Machine Learning for Document Classification of Financial Documents with Focus on Invoices

Maskininlärning för dokumentklassificering av finansiella dokument med fokus på fakturor

Nawar Khalid Saeed

Vårterminen 2022
Examensarbete för civilingenjörsexamen, 30 hp
Huvudområde: Datateknik
Civilingenjörsprogrammet i datateknik
Institutionen för naturvetenskap och teknik, Örebro Universitet.
Handledare: Martin Längkvist, Forskare, Örebro Universitet
Examinator: Farhang Nemati, Universitetslektor, Örebro Universitet

Abstract

Automated document classification is an essential technique that aims to process and manage documents in digital forms. Many companies strive for a text classification methodology that can solve a plethora of problems. One of these problems is classifying and organizing a massive amount of documents based on a set of predefined categories.

This thesis aims to help Medius, a company that works with invoice workflow, to classify their documents into invoices and non-invoices. This has been accomplished by implementing and evaluating various machine learning classification methods in terms of their accuracy and efficiency for the task of financial document classification, where only invoices are of interest. Furthermore, the necessary pre-processing steps for achieving good performance are considered when evaluating the mentioned classification methods.

In this study, two document representation methods "Term Frequency Inverse Document Frequency (TF-IDF) and Doc2Vec" were used to represent the documents as fixed-length vectors. The representation aims to reduce the complexity of the documents and make them easier to handle. In addition, three classification methods have been used to automate the document classification process for invoices. These methods were Logistic Regression, Multinomial Naïve Bayes and Support Vector Machine.

The results from this thesis indicate that all classification methods used TF-IDF, to represent the documents as vectors, give high performance and accuracy. The accuracy of all three classification methods is over 90%, which is the prerequisite for the success of this study. Moreover, Logistic Regression appears to cope with this task very easily, since it classifies the documents more efficiently compared to the other methods. A test of real data flowing into Medius' invoice workflow shows that Logistic Regression is able to correctly classify up to 96% of the data.

In conclusion, the Logistic Regression together with TF-IDF is determined to be the overall most appropriate method out of the other tested methods. In addition, Doc2Vec suffers to provide a good result because the data set is not customized and sufficient for the method to work well.

Keywords

Document classification, Text classification, Invoices, NLP, TF-IDF, Doc2vec, Machine Learning, Logistic Regression, Multinomial Naïve Bayes, Support Vector Machine.

Sammanfattning

Automatiserad dokumentklassificering är en process eller metod som syftar till att bearbeta och hantera dokument i digitala former. Många företag strävar efter en textklassificeringsmetodik som kan lösa olika problem. Ett av dessa problem är att klassificera och organisera ett stort antal dokument baserat på en uppsättning av fördefinierade kategorier.

Detta examensarbete syftar till att hjälpa Medius, vilket är ett företag som arbetar med fakturaarbetsflöde, att klassificera dokumenten som behandlas i deras fakturaarbetsflöde till fakturor och icke-fakturor. Detta har åstadkommits genom att implementera och utvärdera olika klassificeringsmetoder för maskininlärning med avseende på deras noggrannhet och effektivitet för att klassificera finansiella dokument, där endast fakturor är av intresse.

I denna avhandling har två dokumentrepresentationsmetoder "Term Frequency Inverse Document Frequency (TF-IDF) och Doc2Vec" använts för att representera dokumenten som vektorer. Representationen syftar till att minska komplexiteten i dokumenten och göra de lättare att hantera. Dessutom har tre klassificeringsmetoder använts för att automatisera dokumentklassificeringsprocessen för fakturor. Dessa metoder var Logistic Regression, Multinomial Naïve Bayes och Support Vector Machine.

Resultaten från denna avhandling visade att alla klassificeringsmetoder som använde TF-IDF, för att representera dokumenten som vektorer, gav goda resultat i from av prestanda och noggranhet. Noggrannheten för alla tre klassificeringsmetoderna var över 90%, vilket var kravet för att denna studie skulle anses vara lyckad. Dessutom verkade Logistic Regression att ha det lättare att klassificera dokumenten jämfört med andra metoder. Ett test på riktiga data "dokument" som flödar in i Medius fakturaarbetsflöde visade att Logistic Regression lyckades att korrekt klassificera nästan 96% av dokumenten.

Avslutningsvis, fastställdes Logistic Regression tillsammans med TF-IDF som de övergripande och mest lämpliga metoderna att klara av problmet om dokumentklassficering. Dessvärre, kunde Doc2Vec inte ge ett bra resultat p.g.a. datamängden inte var anpassad och tillräcklig för att metoden skulle fungera bra.

Nyckelord

Dokumentklassificering, Textklassificering, Fakturor, NLP, TF-IDF, Doc2vec, Maskininlärning, Logistic Regression, Multinomial Naïve Bayes, Support Vector Machine.

Acknowledgements

I would like to thank all my supervisors at Medius, Robert Szabo, Grzegorz Kępisty and Anders Törnqvist-Svedbergh. I appreciate all the tips and help they gave me during the time period of the thesis. I also want to take the opportunity and thank my supervisor at Örebro University, Martin Längkvist for his support and guidelines. Finally, i would like to thank my family and sweet fiancee for their support through this journey.

Nawar K.Saeed

List of Abbreviations

NLP	Natural Language Processing
TF-IDF	Term Frequency Inverse Document Frequency
OCR	Optical Character Recognition
BOW	Bag of words
LR	Logistic Regression
SVM	support vector machines
MNB	Multinomial Naïve Bayes
SVC	Support Vector Classification
MLP	Multi-layer Perceptron
AI	Artificial Intelligence
TP	True positives
TN	True negatives
FP	False positives
FN	False negatives

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automated document classification, or even called text classification, has been considered an essential task for processing and handling documents in digital forms. To understand the objective of the thesis in its context, the background section aims to introduce the problem domain and the client of this thesis. After that, various sections are introduced that aim to highlight the thesis's motivations, goals, requirements and delimitations. This chapter concludes by highlighting how the remaining chapters will be presented.

1.1 Background

1.1.1 Document Classification

Many companies strive for a text classification methodology that can solve a plethora of problems. One of these problems is categorizing documents into different categories. Document classification is an essential task, especially in information extraction, text retrieval, and question answering [1].

Document classification has been a fascinating area for many researchers and is also used in various applications. In addition, it is considered as one of the primary tasks of Natural Language Processing (NLP), where a set of categories is automatically assigned by a specific text classifier depending on the content of the document. The recent mutations in NLP have encouraged many research and business groups to start developing applications that take advantage of text classification methods, where machine learning is involved [2]. Document classification systems are usually composed of the following phases: Feature extraction, dimensionality reduction, classifier selection, and evaluation, where classifier selection is considered to be the most significant step in document classification [3].

TF-IDF is one of the most used methods that intends to highlight how essential a word is to a document in a collection of documents [4]. To represent a document to machine learning-based methods, it is important to transform the content of the document into a comprehensible format that the classifier methods can understand, i.e., numerical values. TF-IDF and other feature extraction methods are used to represent the documents and transform them into an understandable format. General statistical classifiers such as support vector machines (SVM), Multinomial Naïve Bayes (MNB), or Logistic Regression (LR) can be applied to learn the above-mentioned comprehensible format. Machine learning methods have not been limited to statistical classification methods, but methods for deep learning, such as CNN, can also be applied to represent words and manage documents [5].

1.1.2 Medius AB

Medius AB is a company that works in the field of purchase-to-payment and document automation. Their offices are distributed over various locations in Sweden, but also abroad. Among other things, they work with invoice workflows. As part of the invoice workflow, they work with

different machine learning techniques that intend to extract information from different types of documents, mainly invoice documents.

1.2 Motivation

One of the challenges facing the company is the non-invoice documents that are processed in their workflow. Before the text can be extracted from these files, an Optical Character Recognition (OCR) processing is applied. The OCR intends to convert the text to a machine-readable format. The OCR step itself is time and resource-consuming, which means that this part should only be applied when needed. This is because non-invoice PDFs are often forwarded to the invoice workflow, where they will be manually deleted later.

Moreover, customers using Medius's products often receives large amounts of PDF and image files daily, and the users spend many hours classifying, processing and archiving these PDF files. This basically costs Medius and Medius customers large amount of resources. Utilizing the OCR processing step for non-invoice documents will increase the efficiency of document handling, as it is unnecessary to perform it on PDF and image files that are later deleted during the process. This thesis aims to help Medius optimize their invoice workflow when it comes to classifying these files or documents into invoices and non-invoices by implementing a machine learning-based classifier that can solve this problem and ultimately save the company time and money.

1.3 Research Question

The research question in this thesis is to analyze, implement and evaluate various machine learning classification methods in terms of their accuracy and efficiency for the task of financial document classification, where only invoices are of interest. Furthermore, the necessary pre-processing steps for achieving good performance are considered when evaluating the mentioned classification methods.

1.4 Requirements

As a requirement from Medius, the invoice documents must be classified from other non-invoice documents. Therefore, the method or technique that will be used for the classification should have the following prerequisites:

1. The method's accuracy should be acceptably high, if possible higher than 90%.
2. The method should classify the documents as quickly as possible, if possible less than 1 sec/document.

1.4.1 Additional Requirements

In addition to the requirements listed above, Medius wants to meet other additional requirements. However, these requirements are not mandatory to fulfill. The additional requirements are as follows.

1. Using a test set with invoice and non-invoice documents, the method's precision should be higher than 90%. It means that the mistake of classifying a non-invoice as an invoice should be less than 10%.
2. Using a test set with just images (logos, images of people, animals, objects, etc.), the method's precision should be 99%. It means that the mistake of classifying an image as an invoice should be less than 1%.

1.5 Delimitations

The data set contains multiple classes. Out of these classes, only the invoice and non-invoice classes are of interest. Classification of multiple classes might be an interest for Medius and also can be a base for further research and development.

The data for invoice documents that Medius will provide is already OCR-processed. This thesis will not consider how well and accurately their OCR-engine extracts the text. In addition, other OCR-engines will also be used to extract the text from non-invoice documents. These will not be evaluated either.

Furthermore, no proper check will be performed on the invoice data that Medius will provide to see if, e.g. there are duplicates or other document representations are included in the data set.

1.6 Thesis Outline

This report is organized as follows:

Chapter 2

This chapter presents the essential concepts about text classification that are important to understand. Appropriate methods that can be used for the classification and related work that discusses existing methods will also be presented in this chapter.

Chapter 3

This chapter presents proposed architectures for the methods that will be used. In addition, this chapter highlights the tools and libraries that will be used to implement the proposed methods.

Chapter 4

This chapter explains how the methods have been implemented step by step and in more details.

Chapter 5

This chapter presents the results of the proposed methods for different amounts of test data.

Chapter 6

This chapter discusses the fulfillment of the thesis requirements. The obtained result will also be discussed. A critical analysis of what can be improved will also be addressed.

Chapter 2

Background

To make a computer understand, analyze, manipulate human language and learn from past experiences, methods that can handle these tasks must be implemented. Nowadays, the term Artificial Intelligence (AI) is used in many different contexts. Hence, various subfields of AI have been introduced to meet the challenge in other contexts. NLP and machine learning are two crucial subfields of AI that have been an area of interest for many businesses and research groups. This thesis aims to adopt an NLP and machine learning-based method for classifying documents.

This chapter is divided into various sections. The first sections will introduce NLP in terms of how it is intended to understand and process human language and the process of document classification. The following sections will introduce the fundamental concepts of machine learning and introduce essential algorithms that can be used for text classification and how they can be evaluated. The last section will highlight related work of machine learning-based methods for document classification of financial documents.

2.1 Natural Language Processing

One of the most popular research areas that focuses on how computers can understand and manipulate natural text and speech is NLP. NLP is mainly used to provide computers with the ability to manage natural languages based on how humans interpret and use them. This aims to develop methods that can be used for various tasks such as question answering, machine translation, text summarizing, and text classification systems [6].

Document classification is an area within NLP that ensures assigning a set of predefined classes to documents. This can, according to Ikonomakis et al. [1], be formally described as, if d_i is a document that belongs to a collection of documents D and that $\{c_1, c_2, \cdot, c_n\}$ is the set of classes, then document classification intends to assign a class c_j to the document d_i. It is important to understand how a document's text can be represented to ensure a correct classification process. More details can be found in Section 2.2.3.

2.2 Document Classification

The challenge in the task of document classification is to somehow represent the document so that the specific classification method can understand it. Typically, a text document can be represented as a vector of words. These words are often called vocabulary or features [7]. The problem with such a text representation is that the vector can be huge. The more terms a document has, the bigger the vector will become. This will make the vector have a high dimensionality (many features), which can significantly impact the efficiency and accuracy of the classification method. Hence, a preprocessing step before representing the documents as a vector of words is required. Once the data has been preprocessed, it becomes ready to be represented. This step is called Feature Extraction, which allows efficient data manipulation and representation and is mainly used to decrease the dimensionality of the vector representation. After this step, a vector

representation of words of a certain dimension is constructed. An optional step can be combined with the second step to reduce the dimension of the vector even more. This step is called Feature selection, and the purpose of using this method is to improve the accuracy, efficiency, and scalability of the classifier's method. Typically, it is unnecessary to include all terms when creating the vector as including only important words contributes to increasing the accuracy. However, this step is not mandatory to perform if not needed. After this step, the text becomes ready to be represented as a vector, and the only thing left to do is to transform this vector into a comprehensible format the classifier methods can understand. Finally, the vector becomes ready to be fed into a classifier [8]. Figure 2.1 illustrates the process of document classification.

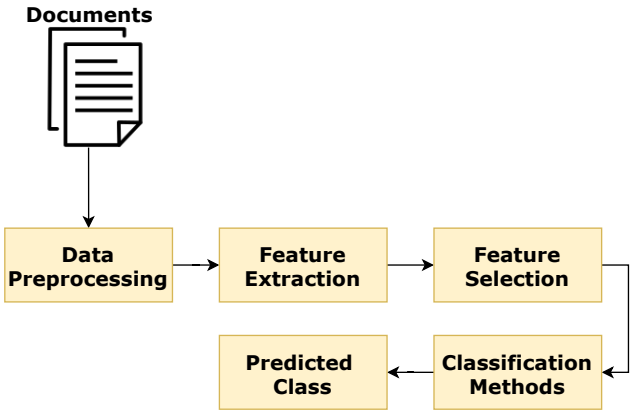


Figure 2.1: The process of document classification.

2.2.1 Data Preprocessing

Machine learning methods that are used to classify text are not able to understand and run on the raw text, therefore the text must be handled in some way. Handling the text is usually called feature preprocessing, and there are many techniques that can be applied to preprocess the text [8]. The preprocessing of text can be done by the following techniques [9]:

Tokenization

This step aims to tokenize extracted text into a sequence of tokens. It can be a difficult task to specify what can be considered as a token, e.g., the term "it's" can be considered as a single token $\{it's\}$ or two tokens $\{it, is\}$.

Filtering

The vector that contains the tokens or the terms will probably contain conjunctions, e.g., as, and, but, etc., and pronouns such as he, she, we, etc. In addition, there may be words that occur rarely and lack statistics in particular. Frequently duplicated words can also arise and must be removed from the vector because they will not have much effect and will not be valuable to the classification process.

Lemmatization

In this step, the verb form is mapped to finite tense and nouns are converted to their singular form. This requires the word form to be known, which is time-consuming. Therefore, stemming methods are often used instead.

Stemming

Stemming methods try to rebuild the form of the word, i.e., change the words to their original state. In other words, a stemmer is an algorithm that deletes words with the same stem but

retains the origin of words, e.g., the words "Running," "Ran," and "Runner" will be replaced by "Run." This will intend to decrease the number of words in the vector.

2.2.2 Feature Selection

One of the significant challenges for text classification is the high dimensionality of the vocabulary. Since most of the words or features are not relevant in most text domains, the classification accuracy is significantly affected. The solution to reducing the high dimensionality is to use what is called Feature Selection to improve the classifier's accuracy and efficiency. The idea behind Feature selection is to select a subset of features with the highest score by finding out feature importance with a predetermined measure [10].

In general, there are two types of Feature Selection methods in machine learning, namely Wrappers and Filters. When it comes to Wrappers, these methods need to train a classifier for each subset of the features, as these methods use the classification accuracy of some classifiers as their evaluation functions. In other words, the subsets will be evaluated by wrappers based on the selected classifier performance. This evaluation is repeated for each subset, which makes them more time-consuming. On the other hand, Filter methods are entirely independent of classifiers, and they select the features based on a specific evaluation measure. Evaluation measures to select features can be applied to the entire subset or rank individual features. However, a filter method can not be chosen completely randomly, but it must meet the task's requirement, e.g., whether it is a classification, regression, or clustering task. The problem domain and the classifier properties should be considered to choose a good feature selection [11].

Method	Mathematical formula
Chi square [12]	$\tilde{\chi}^2 = \frac{1}{d}\sum_{k=1}^{n}\frac{(O_k - E_k)^2}{E_k}$
Pearson Correlation [13]	$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2(y_i - \bar{y})^2}}$

Table 2.1: Common filter methods for feature selection.

2.2.3 Feature Extraction and Document Representation Methods

The text documents, after preprocessing steps (Filtering, Stemming, Lemmtaztion), are usually still high-dimensional unstructured data. This is based on the number of words that the document has. Therefore, it is crucial to have a concise way of representing them. This representation, i.e., transforming the text into another form that facilitates their analysis, can be implemented through a set of phases [14].

The well-known methods for representing the text are presented as follows.

Vector Space Model

One of the most widely used models for document representation is Vector Space Model. This model represents text documents as vectors and has been implemented in information retrieval systems, where documents and queries are represented in n-dimension coordinate systems. Formally, each document can be represented as: $D_j = \{w_{1j}, w_{2j}, ..., w_{tj}\}$, where j is the number of the document and t is the number of the term. A query can be represented as: $Q = \{w_{1q}, w_{2q}, ..., w_{nq}\}$ where n is the term's number for the query [15].

The concept of similarity is fundamental in the Vector Space Model. Usually, the documents are compared with queries using vector operations. This can be achieved by letting the model calculate the similarity between the document D_j and the query Q. Most commonly, Cosine similarity is used to compare these vectors. Mathematically, this can be obtained by the following equation:

$$\cos(\mathbf{D_j}, \mathbf{Q}) = \frac{\mathbf{D_j} \cdot \mathbf{Q}}{\|\mathbf{D_j}\|\|\mathbf{Q}\|} = \frac{\sum_{i=1}^{n} \mathbf{w}_{ij}\mathbf{w}_{iq}}{\sqrt{\sum_{i=1}^{n} (\mathbf{w}_{ij})^2}\sqrt{\sum_{i=1}^{n} (\mathbf{w}_{iq})^2}} \tag{2.1}$$

Equation 2.1, clarifies the similarity between the document D_j and the question Q. In other words, if the similarity is 1, it means that D_j and Q are close to each other or almost the same, otherwise, the closer to 0 the value of the similarity is, the further D_j is from Q [16].

Bag of Words

One of the most straightforward feature extraction methods in document classification is Bag of words (BOW), where the documents represent the individual words as a bag of words, hence the name. In this model, duplicates are considered, while the order and grammar of the words are totally ignored. The semantic relationship between the words is not taken into account by this model, therefore the meaning of words that occur within the same context disappears. Moreover, the occurrence or frequency of words is used to train classifiers [17].

TF-IDF

Term frequency-inverse document frequency TF-IDF is a static measure based on the Vector Space Model. This measure analyses how relevant a word is for a specific document in a collection of documents [18].

The TF-IDF method is calculated from two measurements, namely term frequency (TF) and inverse term frequency (IDF). TF indicates the number of times the term t appears in the document. Since there is a high probability that a word is repeated a lot in long texts than in shorter ones, the term frequency has to be normalized. This is obtained by dividing the term frequency by the number of terms in the document. Mathematically, TF is calculated as follows [19]:

$$TF(\mathbf{t}, \mathbf{D}) = \frac{f(\mathbf{t}, \mathbf{D})}{|\mathbf{D}|} \tag{2.2}$$

$f(t, D)$ indicates the number of times the term $\mathbf{t}$ occurs in the document D and $|D|$ is the number of terms.

The second measure, i.e., IDF, determines how common a word is in the entire collection of documents. This can be obtained by taking the logarithm of the total number of documents $|D|$, divided by the number of documents containing this word DF. Mathematically, this can be described as follows [19]:

$$IDF(\mathbf{t}, \mathbf{N}) = \log(\frac{|\mathbf{N}|}{NF}) \tag{2.3}$$

Finally, TF-IDF can be calculated using the equations as follows [19]:

$$TF - IDF(\mathbf{t}, \mathbf{D}, \mathbf{N}) = TF(\mathbf{t}, \mathbf{D}) \cdot IDF(\mathbf{t}, \mathbf{N}) \tag{2.4}$$

Words Embeddings

TF-IDF solves the problem that the BOW model can not handle, precisely the common terms in the document that have a higher frequency than essential terms. But it still has some weaknesses and limitations. TF-IDF can not determine the semantic relationship between the terms in a document, and the reason for this is that each term is represented independently with its own

index. As a result, the need for methods that can know word equality has increased significantly, and one of these methods is Word Embeddings. This method is designed to capture the semantics between the words [20]. However, it does not mean that TF-IDF is not efficient in solving text classification tasks. In fact, different text classification tasks require optimized methods for what is necessary to solve the task. In other words, classification tasks such as sentiment analysis require procedures that perform sentence analysis and that take into account the semantic relationship between the words [21].

However, to capture the similarity of the words in a semantic vector space, the words are mapped into vectors of real numbers. The purpose of doing this is that words with similar contexts will have similar representative vectors. In other words, vectors containing similar words will be placed closer to each other in a semantic vector space and vice versa [22].

Word2Vec

Word2Vec is a model used to represent text documents based on a neural network concept. The model consists of three layers: input, hidden, and output. In the input layer, all documents are represented. To get the target word in the output layer, feature vectors are required. These vectors are obtained by the weight of the neurons in the hidden layer. These vectors contain, among other things, semantic information about the words, so that words that have similar contexts will have vectors close to each other, and words that are different will have vectors away from each other [23].

Word2Vec method can be applied via two methods, namely Common Bag Of Words (CBOW) and Skip Gram. The main difference between these two methods is that CBOW uses the context for each word, i.e., the words are given next to the target word as input data. The output will be what is predicted as the target word. However, in the Skip-gram model, the target word itself is given as input, and the output is obtained as adjacent words [23].

Doc2Vec

The method was introduced to meet the need to create a vector representation of a collection of words taken together as a document. This method was introduced as an extension to the previously mentioned method Word2Vec, where each word is considered as an independent vector. Specifically, the technique expands Word2Vec from word level to document level but is progressively structured similarly. The only difference regarding how the model is structured is that it has an extra parameter in the input layer representing the document where a given word is located [24].

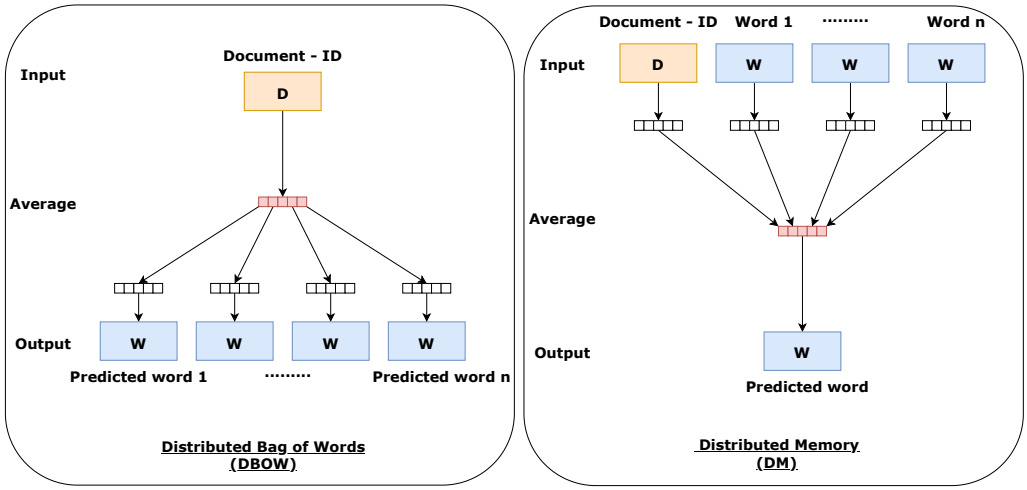


Figure 2.2: Doc2Vec architectures [24].

Like Word2Vec, this model has two methods of representing documents: the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag of Words version of Paragraph Vector (PV-DBOW) as shown in Figure 2.2. The DM model is similar to the CBOW method in Word2Vec. It tries to predict the target word from a collection of adjacent words plus a paragraph ID. The way in which the DM model works is to remember what is missing in the current context. On the other hand, DBOW is similar to the Skip Gram model used for Word2Vec, which is designed to predict the context words from a target word. In the DBOW model, a document-ID is used as input to predict a selection of words from the document randomly instead of using a target word [24].

BERT

BERT is the abbreviation for Bidirectional Encoder Representation from Transformers and is a state-of-the-art deep learning model for NLP tasks based on the Transformer architecture introduced by Devlin et al. [25]. The model can be used to solve various NLP tasks, such as sentiment analysis, text summarizing, and question answering. The primary purpose of this model is to create a pre-trained feature representation of words by considering the word's context from two directions. In other words, it means that feature representation for certain words will depend on both the preceding and following words in a sentence and not just the preceding words. This property, which considers more context to the words, has overcome the encoding of words presented in other methods such as RNNs [25].

As Figure 2.3 shows, the model has three layers. The first layer, which is commonly used in most methods, is called the token embeddings layer and has the purpose of creating a vector representation of the words by using the WordPiece subword tokenization method [26]. The second layer is called Segment Embeddings and has the purpose of handling the input sentences, as the BERT model can handle two input sentences, and distinguish whether a specific token belongs to the first or the second sentence or segment. This is obtained through two tokens. The first token [CLS] is inserted at the beginning of the first sentence, while [SEP] is inserted at the end of each sentence. This means that BERT can, as mentioned before, handle more than one sentence. The third layer is called Position Embedding, and the aim of this layer is basically to remember the order of the tokens by knowing their absolute position within the sentence.

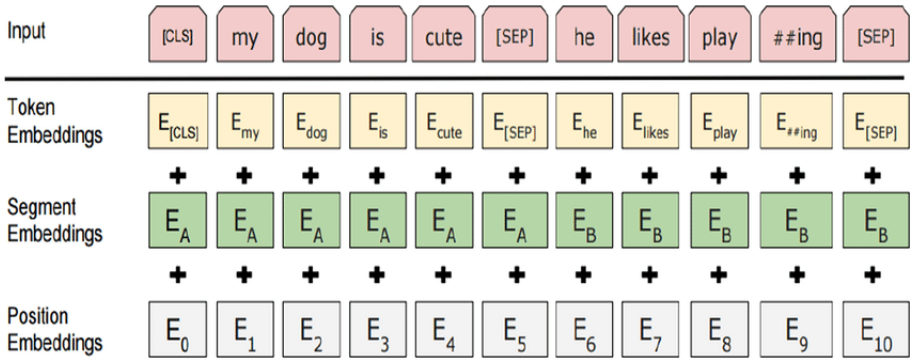


Figure 2.3: Input representations in BERT [25].

The model is pre-trained using two different pre-training tasks [25]. The first pre-training task is called the Masked Language Model (MLM), which randomly masks 15% of the word token. These tokens are masked by replacing them with the [MASK] token instead, which will allow the model to learn to predict the masked tokens bidirectionally. Hence, based on the context provided by non-masked tokens, the initial value of the masked tokens will be predicted by the model. In addition, the model uses another training task in parallel with MLM, namely Next Sentence Prediction NSP. The purpose behind NSP is to teach the model how to predict the following sentence by capturing the relationship between two input sentences.

2.3 General Introduction to Machine Learning

Machine learning can be explained in different ways. However, formulating mathematical models to approximate reality and trying to make future predictions based on these approximations is the fundamental concept behind machine learning. The machine becomes smarter through more experience and learning of the input data it receives, which means that more different tasks can be solved by machine learning [27]. The machine learning algorithms are divided into different categories such as supervised, semi-supervised, unsupervised, etc. Since this thesis will only focus on solving a supervised classification problem, only supervised learning will be introduced.

2.3.1 Supervised Learning

The learning that is based on iterative predictions that the learning methods make based on labeled training data is called supervised learning. For a classification problem, the learning method learns from a set of labeled data in the training set to be able to identify unlabeled data in the test set [28]. It can be, e.g., Spam detection where an email message is identified as spam or non-spam or classifying financial documents into invoice documents and non-invoice documents.

2.3.2 Training and Test Sets

The training set for a supervised learning method usually consists of n number of ordered pairs $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where each x_i is related to a set measure for a specific sample data point and y_i is associated with the label or category of this sample data point [28]. For example, x_i can be a single document. The corresponding y_i is the classification of what the document can be, e.g., invoice or non-invoice.

The test set contains another set of unlabeled data, where the goal is to evaluate the final model after the training step. Usually, the test data set is used to obtain the performance characteristics such as accuracy, sensitivity, F-measure, etc [28]. Figure 2.4 illustrates how a data set is usually split.

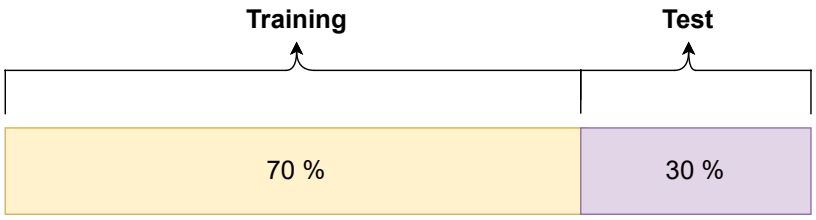


Figure 2.4: Splitting data into training and test sets.

2.4 Machine Learning-based Text Classification Methods

Nowadays, various methods have succeeded well with text classification problems.The decision to choose the most appropriate classification method must depend on the different characteristics of the text attributes [29]. In this section, some text classification methods will be represented.

2.4.1 Bayesian Classifiers

Bayesian classifiers are a collection method that can classify text by probabilistic methods. These methods classify the text based on the posterior probability that the documents belong to a certain class based on the word's presence in the documents. One known method that belongs to these methods is the Naïve Bayes Classifier [30].

Two models can be used for Naïve Bayes classification, where both methods ignore the actual position of the words in the document and assume a text-vector representation, e.g., Bag of word models [29]. These models are **Multinomial** model and **Multivariate Bernoulli** model. The only difference between these two models is that the Multinomial model takes into account the feature vector representation where the assumption of the word frequency is used, while the Multivariate Bernoulli model is used when the word frequency is of non-significance and the assumption that word features are assumed to be binary [31].

Since the text document will be represented in this thesis as a vector of features where the frequency of the words is essential, only the Multinomial model will be represented.

Multinomial Naïve Bayes

This model treats the documents as a set of words with frequencies linked to each word. Let's assume that C represents a set of classes and N represents the size set of words. The highest probability that the test document d_i belongs to a class can be defined by Bayes' rule as follows [32]:

$$P(c|d_i) = \frac{P(c)P(d_i|c)}{P(d_i)}, \qquad c \in C \tag{2.5}$$

$P(c)$ is the class prior and can be estimated by dividing the number of documents belonging to class c by the total number of documents. $P(d_i)$ is the normalization factor. $P(d_i|c)$ is the probability of obtaining document d_i in class c and can be calculated as follows:

$$P(d_i|c) = (\sum_n f_{ni})! \prod \frac{P(w_n|c)^{f_{ni}}}{f_{ni}!} \tag{2.6}$$

f_{ni} is the number of words in the test document t_i. In the numerator it is $P(w_n|c)^{f_{ni}}$ which represents the probability that the word n given the class c. However, according to [32] Equation 2.6 can be refactored to be as:

$$P(d_i|c) = \alpha \prod p(w_n|c)^{f_{ni}} \tag{2.7}$$

2.4.2 Support Vector Machine

SVM is a supervised machine learning method for binary classification and regression analysis. This method attempts to partition the data using linear and non-linear boundaries between different classes. In other words, the method tries to classify different classes by determining the optimal boundaries between these classes. The method receives, in the first phase, the data in pairs $(e_1, l_1), (e_1, l_1), ..., (e_n, l_n)$ where e_i is an element that is linked with a label l_i. In the second phase, the method becomes a "black box" that provides an output [33].

According to the illustration, as shown in Figure 2.5, a decision boundary hyperplane is constructed, which divides the space into two sub-spaces, where the margin between the two classes is maximum. One subspace contains vectors belonging to a class, while the other subspace contains vectors belonging to the other class [34].

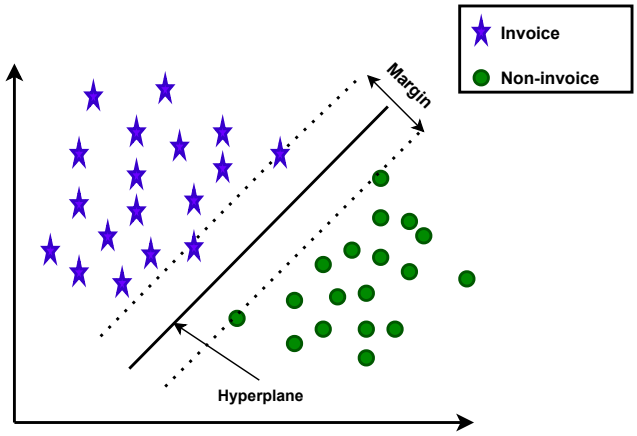


Figure 2.5: Illustration of SVM classification of two classes.

2.4.3 Multi-layer Perceptron

One of the simplest models based on the neural network concept is Multi-layer Perceptron (MLP). This model consists of an input layer, a hidden layer with an activation function, and an output layer, according to Figure 2.6. The data, which are vectors that contain information about the words, flow in one direction through the input layer and then pass through the hidden neurons and finally to the output layer. In this method, all the neurons in one layer are connected to all the neurons in the next layer with a certain weight. Within the input layer, all data has a predefined weight that will be fed to the hidden layer. In the hidden layer, they can consist of several layers if needed, there is what is called an activation function which can be a logistic function, hyperbolic tangent, or Rectified Linear Unit. The activation function calculates the output signal "weight" and passes it on to the next layer of nodes in the network. The process is repeated until the signal reaches the output layer [35] [36].

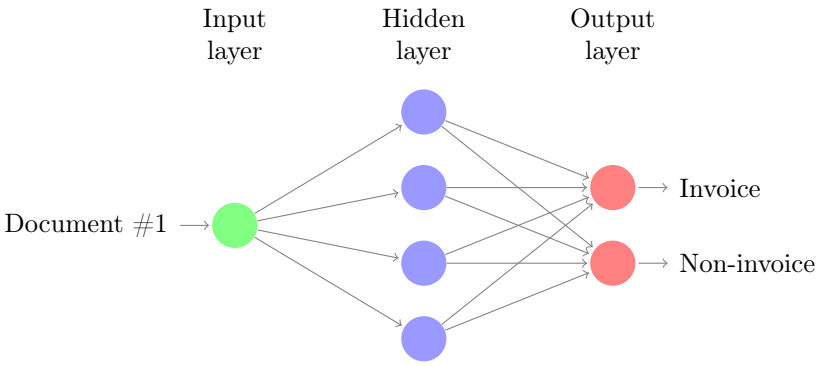


Figure 2.6: Simple illustration of MLP.

The training process for MLP is usually done using a method called BackPropagation [37]. Within each epoch, the output value of the training data is calculated and then compared with the correct value by finding out the error between them. The error calculated by a loss function will be sent back from the output layer to the input layer to adjust the input weights [38].

2.4.4 Logistic Regression

One of the most well-known and used supervised machine learning algorithms that are mainly used to indicate the probability of a particular category is LR. In a very simple manner, given a document d represented by a set of features $x_1, x_2, ..., x_n$, the probability that $p(c_j|d)$ of d to belong to a class c_j is calculated as follows:

$$P(c_j|d) = \sigma(\sum_{i=1}^{n} x_i \times w_i) = \sigma(w^d x) \tag{2.8}$$

In other words, the probability is obtained by linearly combining the set features with a log function. Hence, the $\sigma(x)$ or even called the sigmoid function is used in the case of a binary classification problem. $\sigma(x) = \frac{1}{1+e^{-}w^d x}$ and $w^d x$ is the decision boundary.

Since the target variables or the categories to predict can only have two possible types, it is easy to model a relationship between the binary target variable and the predictor variables. This connection can be obtained using the sigmoid function. Figure 2.7 below illustrates what the sigmoid curve looks like and how it is used to predict the target class.

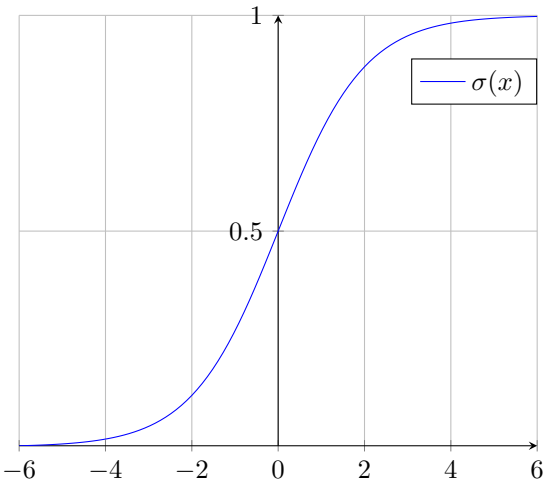


Figure 2.7: Sigmoid function.

2.5 Validation Measurements

An essential part of any project based on machine learning is to evaluate the models by providing some feedback about their performance. An evaluation metric can specify the predictive model's performance and provide helpful feedback. In machine learning, there are several evaluation measures to consider where all these measures have a common property. More specifically, all of these fall within the range [0,1], where the value near 1 indicates that the model performs well, and the value near 0 indicates that the model performs poorly [39]. Below, the most common metrics to consider are presented.
Before presenting these metrics in more detail, there are a few terms to be aware of:

- True positives (TP): is the case where the model predicts the positive class correctly.
- True negatives (TN): is the case where the model predicts the negative class correctly.
- False positives (FP): is the case where the model predicts the positive class incorrectly.
- False negatives (FN): is the case where the model predicts the negative class incorrectly.

2.5.1 Accuracy

One of the most straightforward metrics used to determine how many of the predictions are correct out of the total amount of data is Accuracy. In other words, accuracy is the fraction of predictions that the model gave. This metric can be calculated as follows:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2.9}$$

However, the result of this metric can often be misleading, especially in the case of an unbalanced data set, as the model decides that the best thing to do is always to predict the class with the highest distribution. Hence, this metric can not be considered the best choice for evaluating the model.

2.5.2 Recall

Another metric that can be used in the case of the high cost of false negatives is the Recall metric. This metric illustrates the proportion of true positives that were correctly identified. Formally, this metric can be calculated as follows:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.10}$$

2.5.3 Precision

Unlike the Recall metric, the Precision metric is used in the case of assigning a high cost to false positives. It tries to figure out how often it is correct when it predicts positively, or in other words, it lists how large a proportion of positive identifications are correct. Formally, this measure can be calculated as follows:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.11}$$

Precision and Recall metrics can be beneficial in the case of imbalanced data sets. They can undeniably replace the accuracy metric as they can list out more valuable information than what accuracy metric can.

2.5.4 F1-Score

Again, it isn't easy to evaluate the model's performance in the case of imbalanced data sets. An important metric that is very useful in this case is F1-Score or also called F1-measure. This metric indicates the harmonic mean of recall and precision metrics. This metric is convenient and valuable in the case of imbalanced data sets because it captures the balance between precision and recall. Formally, this metric can be calculated as follows:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.12}$$

2.5.5 Confusion Matrix

An also important metric that associates predicted classes with actual classes is called the Confusion Matrix. This metric is based on a helpful table that shows the performance of a set of data where all actual values are known. In addition, it describes where and how misclassifications occur, i.e., which classes have been confused with each other. The figure below illustrates that each row represents the essential truth of a specific class and each column indicates the classes predicted by the model.

	Positive	**Negative**
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Figure 2.8: Simple illustration of confusion matrix

2.5.6 ROC Curves

It is pretty good to highlight a helpful evaluation method that is represented as a graph showing a model's performance beyond all classification thresholds. This evaluation measure is called the ROC curve and is based on two parameters: True Positive Rate (TPR) "Recall" and False Positive Rate. False Positive Rate (FPR) can be calculated as follows:

$$FPR = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.13}$$

ROC curve plots TPR and FPR at different thresholds. A measure called Area Under Curve (AUC) can be used to describe the performance across all possible classification thresholds. This measure is typically used for binary classification problems. This measure aims to calculate the entire two-dimensional area under the ROC curve. AUC has a range of [0, 1], and the closer the value of the area to 1, the better the performance of the model [39] [40].

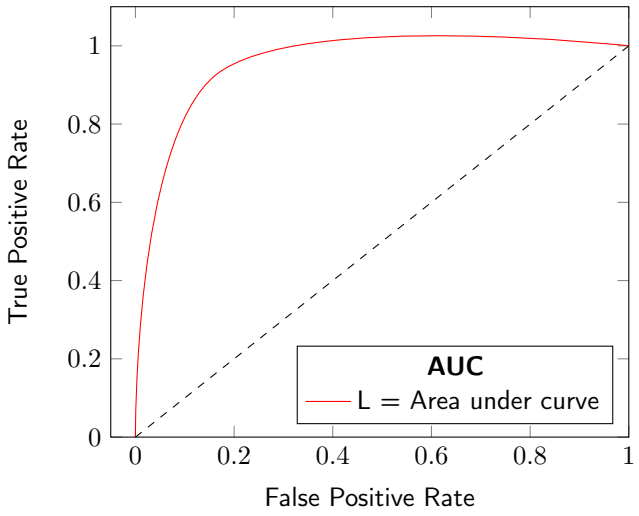


Figure 2.9: Simple illustration of ROC-Curve.

2.5.7 Learning Curves

The Learning-Curves are used, among other things, to determine whether more data is needed to train the classification methods or not. This is achieved by understanding how the data affects the method's performance. In addition, it can be used to indicate error due to variance and error due to bias, i.e., if the model suffers from overfitting or underfitting [41].

2.6 Related Work in Financial Documents Classification

Automated text classification has always received considerable attention, even though many proposed approaches. Although state-of-the-art models are efficient and handle text classification tasks well, there is still a lack of application to cover all kinds of text classification domains, especially when it comes to unstructured documents such as financial documents [42]. Other approaches have also been introduced, such as template-based classification methods, where the template for an invoice is either calculated and represented by a set of layout properties or assigns clusters with similar attributes. In this way, an invoice can be recognized and categorized. The problem with such an approach is that it is constantly expanding in the number of clusters or templates needed [43]. More about it is discussed in Chapter 6.

Some of the related works for automated text classification for financial documents are described as follows.

2.6.1 Existing Classification Methods for Unstructured Documents

A couple of models that consider the text as a sequence of words are the RNN models. RNN stands for Recurrent Neural Network and is designed to capture text structures and word dependencies for text classification tasks. One of the most well-known architectures for RNN is Long Short-Term Memory (LSTM), designed to solve the problem of learning long-distance correlations in a sequence [44]. A method that combines the LSTM, to obtain the representation of sentences and a CNN model [45], to extract a sequence of phrase representations, has been developed. This method is called C-LSTM and has the property of capturing semantic relations of sentences and achieving good performance on sentiment classification and question classification tasks [46].

An object detection model and a Regional Proposal model were combined to train a detection network [47]. This model has been adapted to detect the category of books [48]. This model is known as the Faster Region-based Convolutional Neural Network R-CNN and is developed to predict the bounding boxes that belong to the books and detect their categories.

Another approach has also been developed to identify and classify invoices presented in this study [49]. The model is presented as a conventional template-based method that applies slope removal and rotation angle reduction for invoice images. It might be worth mentioning that the model uses only twenty template images to match the regions in text blocks so that valuable information can be extracted.

Since invoices are considered examples of short text [50], Naseem et al. [51] proposed an extended meta-embedding method for sentiment analysis of the short text that combines sentiment lexicon, features provided by word embedding, and part of speech tagging to a compound vector. Furthermore, the vector is fed to a Bi-LSTM [52] using an attention network. This approach is limited due to the complexity of language. Hence each association vector provides a specific understanding of the language.

Another image-based approach that aims to train a lightweight deep learning model based on Resnet-18 [53] architecture has been presented by Wang et al. [54]. The model is designed to operate on a small data set, which can be considered the most substantial part of the model. The data set consists of 1200 images with three categories: value-added tax, train, and taxi invoices, which probably have similar layouts. Furthermore, the article discusses that there are many types of invoices in the market, and additional invoice categories can further optimize the model and meet the current needs.

Chapter 3

Architectures and Tools

This chapter aims to introduce the architectures of the methods to provide a better understanding of how these methods intends to work. In addition, this chapter will present the tools used to implement these methods.

3.1 System Architecture

Since the classification methods will be accomplished over supervised machine learning, the classifiers will be implemented according to Figure 3.1 below.
A text classifier is usually implemented in several steps. First, labeled documents are represented as vectors of a certain length. The documents will then be preprocessed to remove unwanted words, punctuation, and strange characters. Furthermore, the documents (the whole data set) will be divided into two sets, namely the training set and the test set. The training set will be fed into the classifiers to train them, while the test set will be used to evaluate and predict the results. The pre-trained models "classifiers" will be saved locally to be used afterward to classify unseen or unlabeled documents, as Figure 3.3 shows.

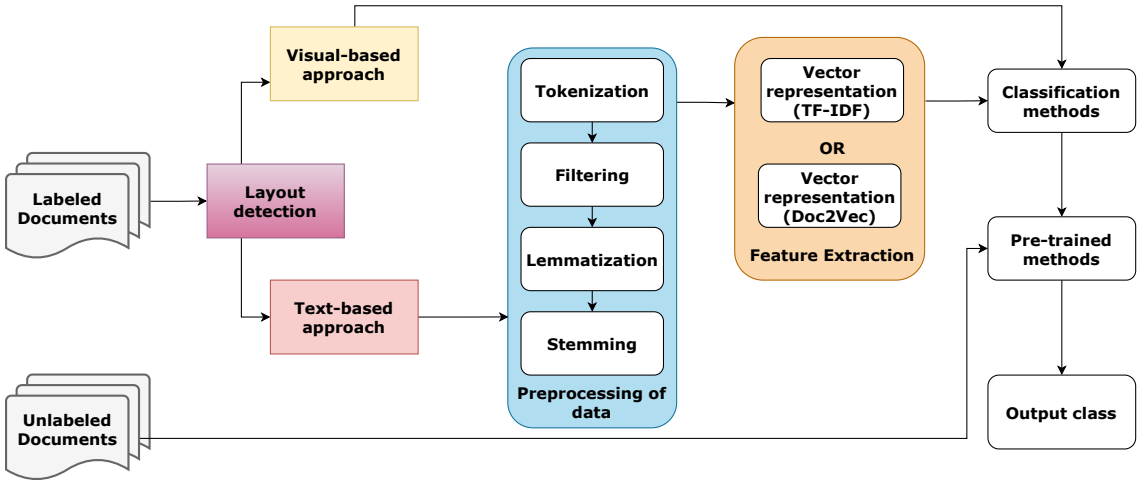


Figure 3.1: An overview of the proposed system architecture.

A more detailed description of the method is presented in Chapter 4.

3.1.1 Text Extraction

The data set that will be processed contains both text (financial documents and other types of documents) and images (advertising, company logos, and white blank pages). From the beginning, the idea was to execute a Layout detection method that detects the layout of these

documents and then predicts whether a visual approach or text-based approach is suitable to use. The first assumption was to adapt some visual approach to solving the classification of invoice documents, as Figure 3.1 shows. The visual approach is about analyzing the visual structure of the document and template matching without taking into account the content of the text. However, this approach is not the best choice for classifying unstructured documents, including invoices, as mentioned in related work Section 2.6. The invoices have independent layouts and are very different from each other. Therefore, a better way to solve the problem of classifying invoice documents is by implementing a text-based approach. This means that no layout detection method will be implemented. Figure 3.2 shows how the proposed architecture for the text-based approach looks like.

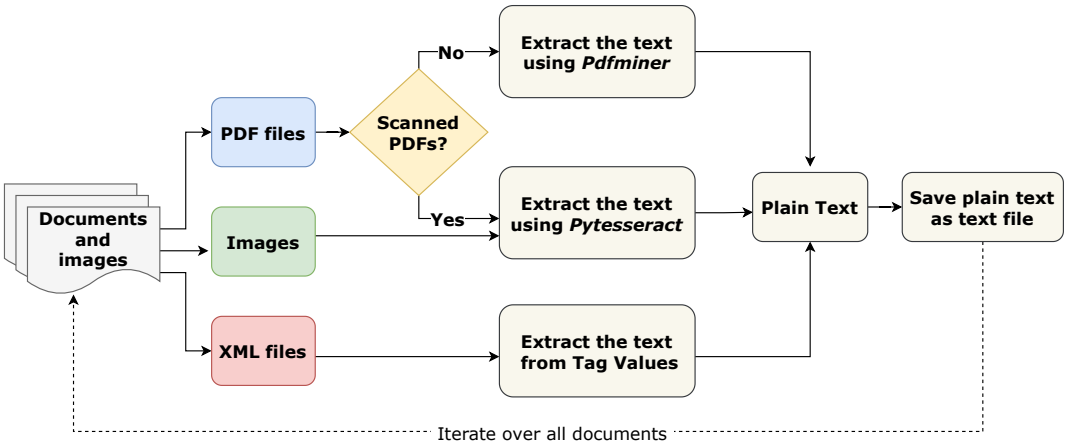


Figure 3.2: An overview of the proposed text extraction architecture.

3.1.2 Training of Classification Methods

In order to train the classification methods, it is essential that all the words are converted to numeric values. For that, encoded document vectors must be created and fed into the classifiers in order to train them. More about this is discussed in Chapter 4. Figure 3.3 intends to show the proposed architecture for the learning process of the classifiers.

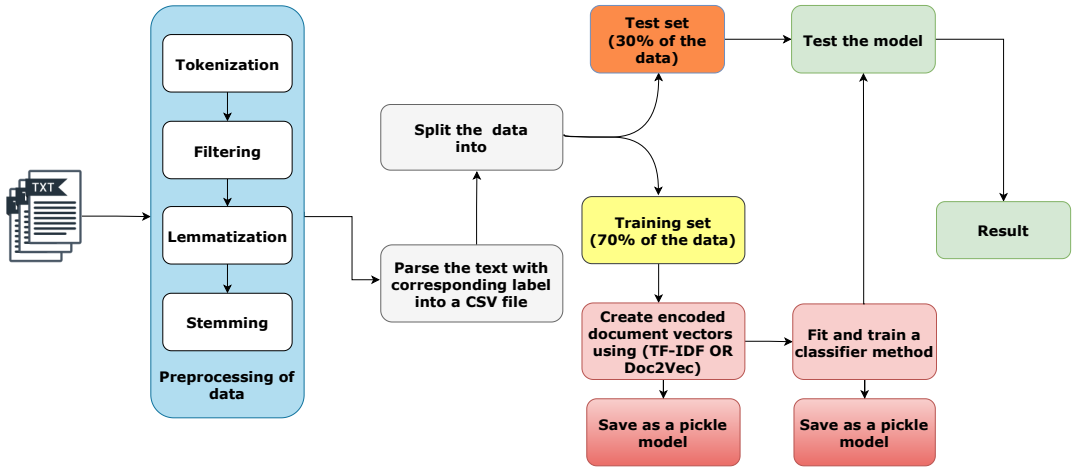


Figure 3.3: An overview of the proposed architecture for the training process of the classifiers.

3.1.3 Classification of New Documents

After encoded document vectors and pre-trained methods are saved locally, unseen documents can be classified and moved to their specific folder. The process of classifying an unseen document is illustrated in Figure 3.4.

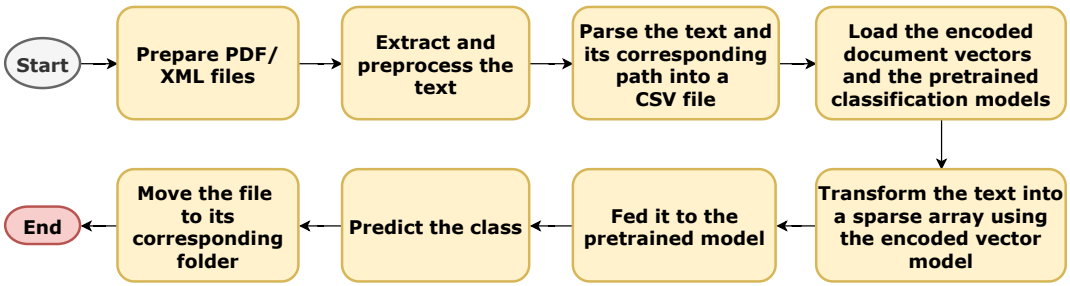


Figure 3.4: The process of classifying a new document.

3.2 Tools

3.2.1 Hardware

The development of the system has taken place in a virtual machine provided by Medius. The reason for using a VM is to preserve the data sets, which are sensitive financial documents that Medius receives from their suppliers. More detailed specifications of the VM are described in Table 3.1.

Components	Description
Name	nawsae-VM
CPU	Intel® Xeon® Platinum 8171M, 2.60GHz
Ram	8 GB 2133MHz
GPU	Unknown
Operating system	Windows server 2019 version 1809

Table 3.1: VM-specifications.

3.2.2 Development Environment

PyCharm Community Edition 2021.3.2

Pycharm, an IDE from JetBrains, is used for the project. The reason for choosing PyCharm as an IDE is because it is a Python-specific development environment and can facilitate various things such as syntax, fetching documentation of a specific method, and so on. Another reason for choosing it is related to a personal habit.

3.3 Programming Language and Program Libraries

3.3.1 Programming Language

Python

When it comes to programming languages, Python version 3.8 was used to build and evaluate all of the methods. The reason for using Python as a programming language is based on simple user-friendliness. Moreover, there are many modules and guides that can be used to do most things in machine learning. Furthermore, it is widely used in research [55].

3.3.2 Libraries and External Functions

Pdfminer

Pdfminer is a tool that aims to extract text from PDF documents. This tool offers many valuable features, such as getting the exact position of the text on a specific page. It can also specify fonts or lines. Moreover, the tool's main purpose is converting PDF documents to other text formats. Despite all these advantages that this tool offers, it also has disadvantages. This tool does not work for scanned PDF documents that do not have text-layers [56].

Pytesseract

Pytesseract is a python optical character recognition (OCR) wrapper to invoke Tesseract from Python. This tool, unlike Pdfminer, can read and extract embedded text, i.e., it can extract text from the PDF document that does not have a text-layer. It has many valuable features, e.g., detection of text language, specific tags and words. It can also be used to extract the position of text-fields in a document [57].

NLTK

Natural Language Toolkit is a Python library that is mainly used to handle natural language processing methods NLP. The library offers several methods that are primarily performed within the preprocessing step, such as removing unnecessary stop words for further analysis. It provides text stemming that converts the words to their original stem by removing prefixes and suffixes, e.g., ing, s, es, etc. It also offers Lemmatization, POS Tagging, and much helpful text processing steps [58].

Scikit-learn

When it comes to the implementation of the classification models step, which is considered as an essential step in any machine learning-based system, a library that integrates a wide range of state-of-the-art machine learning algorithms and offers good performance, user-friendliness, documentation, and API consistency is needed. One such library that provides these features is Scikit-learn. Scikit-learn is widely used in the scientific Python community, and a massive area of machine learning applications is supported. It offers one of the widely-used methods of text vectorization, namely TF-IDF [59].

Other Common Libraries

A Python library called Pandas has been used to facilitate the work with data sets and provide fundamental functions for implementing static models. This library offers a wide range of possibilities for data format. Good documentation, visualization, and an extensive range of static calculations have made Pandas an unbeatable library for data preparation [60].

When it comes to data visualization and graphical plotting, the Matplotlib library has been used [61].

Chapter 4

Materials and Methods

This chapter presents the methodology that is implemented in this thesis. Firstly, the data set that has been processed and used is presented. Furthermore, the preprocessing methodology for how the data set is prepared will be presented. Moreover, this chapter explains how the data is converted to numerical values and fed into the classification methods. Finally, the construction and evaluation of the classifiers and how the entire system is presented.

4.1 The Dataset

The first version of the data set that Medius has provided is categorized into four categories: Financial Documents, Images, Terms and Conditions, and Other Documents. These four categories contain the following documents:

- Financial Documents: These documents are mainly invoices.
- Images: These are white pages and images representing something else than a document. This category is the easiest category to be classified.
- Terms Conditions: These are terms and conditions that have different layouts from the financial documents. This category is harder to classify than Images.
- Other Documents: These are probably not financial documents but are very similar in content and layout. This category is the hardest to classify.

However, the distribution of the first version of the data set does not have the required variation. The Financial Documents were about 600 documents, and the other documents were about 200 documents. Due to that, another data set has been provided by Medius. The new data set consists of both sorted documents and mixed documents. The sorted documents were financial documents, and they have been labeled as invoice documents.
When it comes to the mixed document category, the idea was to use them as non-invoice documents. However, it has been difficult to manually classify the mixed data set and pick up approximately 145,000 non-invoice documents. Hence, a public data set called RVL-CDIP has been used [62]. It is distributed into 16 classes. Some of these classes were selected to represent non-invoice documents such as forms, advertisements, specifications, etc. The final data set used in the thesis is presented in Table 4.1.

Documents	**Amounts**	**Format**	**Provider**
Financial(invoices)	144 331	XML	Medius
Other (non-invoices)	144 541	TIF	RVL-CDIP
Total	288 872		

Table 4.1: The data set that will be used in this thesis.

One important thing to mention about the final data set is the language variation among the documents. Approximately 38 different languages have been detected, most of which are English followed by Swedish. There is a very unbalanced distribution regarding how these languages are distributed among the documents. Figure 4.1 shows the five most significant detected languages.

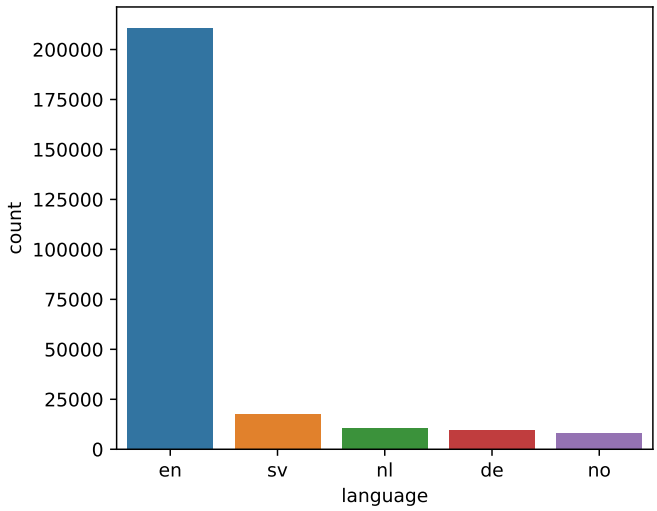


Figure 4.1: The distribution of detected languages among the documents.

As more data has been provided, the first version of the data set will be used as real data flowing into Medius' workflow. The classification method will be evaluated based on this version of data. More about it can be found in Section 5.2.

4.1.1 Preparing the Dataset

The data used in this thesis is arranged partly by Medius and partly by using the public data set RVL-CDIP, as described in Section 4.1. As a reminder, the task was to classify the data into invoices and non-invoices, hence the data was split into two folders. The first category of the data Financial Documents "invoices" was saved in one folder called invoice, and the other categories were saved in another folder called noninvoice. These folders will mainly be used to label the documents.

4.1.2 Text Extraction

In this step, a method for extracting the text from the documents has been implemented. This has been performed in different phases, as the files were scanned pdfs "without text-layer" and electronic pdfs "with text-layer" as well as other representations such as tif which is an image format. First, Pdfminer was used to iterate through all the PDF files to extract the text from them. Each PDF file has been handled separately for each folder, and the extracted text has been saved with the same name as a text file.

The text extraction using Pdfminer has been applied to all files in the invoice and non-invoice folders. However, many text files were empty, as Pdfminer failed to extract the text from scanned PDF files. Pytesseract came into use here, as all empty files were processed by it. The names of all empty text files were saved in a list. That list will be used to compare the names of the PDF files. Pytesseract will handle matched files to extract their text and replace the empty text files. The reason for using two libraries to extract the text was that Pdfminer was relatively fast

compared to Pytesseract. However, Pytesseract has been mostly used to extract the text from the RVL-CDIP data set that has been used to represent all non-invoice documents.

It is worth mentioning that the Medius has its own OCR-engine that extracts the text from the PDF files and saves it as XML files. All documents representing invoices, which Medius has provided, have been OCR-processed and saved as XML files. For that, a method has been created to iterate through all XML files, in the same way as mentioned above. The text has been extracted from specific tags in the XML file saved as a text file.

4.1.3 Text Preprocessing

It is crucial to preprocess the text to improve the model's performance. This has been done by converting all the terms "words" to lowercase and removing unwanted special characters, punctuation, and line breaks. Lemmatization and Stemming method has been also applied to get the root or the base form of the words. Furthermore, stopwords were also removed from the corpus. This will help the models to only take important words into account. After that, the text were parsed into a CSV file (comma-separated values). The CSV file has two columns, namely **docs** which contains the preprocessed text, and **label** which has been used to label the corresponding text. The "invoice" and "noninvoice" folders were used for labeling. This means that all documents in the invoice folder will be labeled as invoice and vice versa for the second folder. Finally, the words were tokenized and a vector of tokens has been created. Converting the text to a vector of tokens means that each token can be represented by a numeric value corresponding to its weight. All these steps were done from scratch and with the help of the NLTK library.

4.1.4 Representation of the Documents

As previously stated, machine learning methods require numerical values as input data, since they cannot handle strings. The words were converted into numeric values corresponding to their weights, which is the comprehensible format that the classifiers can understand. The selected method that converts the tokens "words" to numeric values was TF-IDF. One strong propriety of this method is finding out how characteristic a term was for its class. This was obtained by weighting down existing terms (the, of, is, etc.). The vector representation was created using TfidfVectorizer from scikit-learn, which has a number of parameters that specify the transformation of the string features into numeric feature vectors in the range of [0, 1]. However, encoded document vectors that are required by the classification methods need to be created. This is done by a function that aims to learn a vocabulary from all documents and then encode each of the documents as a vector. A sparse matrix with normalized scores between 0 and 1 will be returned, which can be used by the classification methods.

Another method that has also been implemented is Doc2Vec which is designed for vector representation of entire documents. The method was constructed by a Python library called Gensim [63]. The architecture used to create a vector representation of the documents is DBOW, which has been presented in Section 2.2.3. For this method, TaggedDocuments were created, which are what the model takes as input. TaggedDocuments are structured as a list of tokens with a corresponding tag. The model has been initiated with a vector size of 100 words, where the minimum number of words frequency is set to 2. This avoids having unwanted features and also give high-frequency features more weight. In addition, to give the model more accuracy, the number of iterations over the training corpus is set to 20. This can be increased to further increase the model accuracy, but will require more resources. Finally, to get the encoded document vectors, the infer_vector() function has been used for that purpose. This contains numerical scores that can be used in the same way by the classification methods.

4.2 Classification Methods

The implementation of all classification methods has been performed using the Scikit-learn library. When it comes to LR, this method has been implemented using LogisticRegression from

sklearn.linear_model. For the implementation of Multinomial Naïve Bayes, the MultinomialNB algorithm from sklearn.naive_bayes has been used. Finally, SVM has been implemented using SVC with linear kernel from sklearn.svm. Various parameters have been tested when these methods have been constructed. The best parameters that have been selected for the methods have been presented in Section 5.2.

4.2.1 Methods Implementation

Once the encoded document vectors have been created, the classifiers will be ready to be trained. All classification methods have been implemented by using the machine learning library scikit-learn, as discussed above. The data has been split into 70% training set and 30% as a test set, which is a common practice in machine learning [64]. The training data has been used to train the methods, and test data has been used to predict the results and indicate the methods' performance and accuracy. After the models have been trained with essential terms from the training set, that allows them to distinguish between the documents, they will be serialized and saved as .pickle files. This has been obtained by using the pickle library. Saved models may, later on, be deserialized to make new predictions.The entire procedure is represented in Algorithm 1 below.

Algorithm 1 Training process of the classifiers.

```
Input: Dataset (text, label)
Output1: Trained classifiers clf
CSV_file ← ∅
for each (text, label) ∈ dataset do
    cleantext ← preprocess_data(text)
    CSV_file ← parse_to_CSV(cleantext, label)
end for
data = read(CSV_file)
vectorizing_method ← userinput
if vectorizing_method == "tfidf" then
    x_train, x_test, y_train, y_test = train_ test_split(data, test_size = 0.3)
    vectorizer = TfidfVectorizer()
    vectorizer.fit_transform(x_train)
    vectorizer.transform(x_test)
    save(vectorizer)
    clf.train(x_train, x_test, y_train, y_test)
    save(clf)
end if
if vectorizing_method == "doc2vec" then
    trainTag ← TaggedDocument()
    testTag ← TaggedDocument()
    doc2vec ← Doc2Vec()
    doc2vec.build(trainTag)
    save(doc2vec)
    y_train, x_train = vectorize(model, trainTag)
    y_test, x_test = vectorize(model, testTag)
    clf.train(x_train, x_test, y_train, y_test)
    save(clf)
end if
```

4.2.2 Hyper-parameters Optimization

All classification methods have a number of parameters that can be adjusted to achieve the desired result. It can be challenging to predict what value these parameters should be set in order to increase the model's classification accuracy. For e.g., in Multinomial Naïve Bayes, the hyper-parameters are the smoothing parameter alpha and fit_prior [65]. Manually trying different combinations of values on these two parameters is inefficient and time-consuming, especially for other methods that have much more parameters. Therefore, it is better to develop a better way to optimize and find the best combination of these hyper-parameters. For that, $GridSearch()$ from scikit-learn has been used. This function performs $k - fold\ cross - validation$ to improve the performance of the classification method and provide more reliable results in the evaluation. The principle of $k - fold\ cross - validation$ is based on splitting the learning process k times with different amounts of the initial data as training set and test set. Then, the training set is divided into k parts, where $k - 1$ fold is used for training, and 1 fold is used for testing and validation. While the classification method is trained by the training set ($k - 1$ folds), the

method's performance will be recorded until each of the k-fold got used for testing. Finally, the mean value of the error rate obtained by the fitted model on the k^{th} will be calculated. The mean and standard deviation values, which are used to estimate the error rate, are calculated for each of the k folds, and then their most optimal scores will be selected, indicating that the best hyper-parameters have been chosen for the model. In this thesis, 5-fold cross-validation has been used to tune the hyper-parameters of the methods.

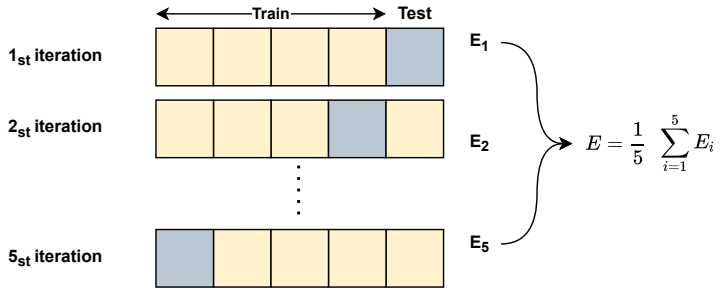


Figure 4.2: Simple illustration of 5-fold cross-validation.

4.2.3 Methods Evaluation

In order to have detailed information on how the classification methods behave in terms of performance and accuracy, it is essential to use evaluation matrices. All of the matrices presented in Section 2.5 have been used to evaluate the performance and accuracy of the methods. Recall, F1 score, accuracy, precision, confusion matrix, ROC-Curve and Learning-Curve have been implemented using the scikit-learn library.

4.2.4 Classification of New Documents

For the last step, a method that categorizes unseen documents into their specific folders has been implemented. This step assumed that Medius would use its own OCR-engine to extract the text from the documents. Hence, the data (the documents) to be classified will be XML files, as Medius's OCR will save extracted text as XML files.

For each XML file, the following will be performed. First, the text will be extracted from Value Tags in the XML files. After that, the text will be preprocessed where stopwords will be deleted, words are converted to lowercase, etc. After that, each preprocessed text and the name of the corresponding XML file will be saved in a CSV file. This means that the CSV file will have two columns, one for the preprocessed text and one for the name of the corresponding file. The purpose of saving the file name is to move the file to its specific directory after it's class has been predicted. In other words, if a text has been predicted as an invoice, the name of the file which this text belongs will be used to move the file to the specific folder where invoice documents are saved. Figure 3.4 shows the process of classifying a new document.

Chapter 5

Experimental Results

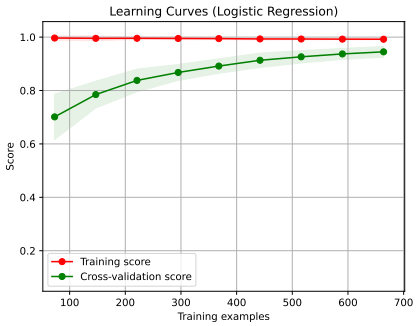
This chapter presents the results from the tests that have been performed on the selected classification methods. Firstly, the performance of how the methods behave and how much data is needed or appropriate to use will be presented. This will be done by using Learning curves. Based on what Learning Curves indicates, the data will be selected for training and testing of the methods. However, the workflow for how the tests will be performed will not be the same, as different data sets and parameters will be used. Various tests will be performed with different data sets, that vary in size, and vectorizer methods, i.e., TF-IDF and Doc2Vec. Finally, a test for the best-performing methods will be performed on a real data set.

5.1 Test Setup

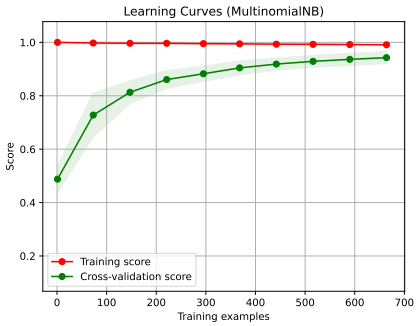
The tests have started by using TF-IDF as a word vectorizer. For the first two tests below, all default parameters of the *TfidfVectoriser* form scikit-learn have been used. The only two parameters that have been manipulated are tokenizer and n_grams. A tokenizer function created via the preprocessing step has been used for the tokenizer parameter. In addition, n_grams was set to (1,2) to include essential terms, e.g., invoice date will be tokenized as ['invoice', 'date', 'invoice date']

5.1.1 Learning Curves

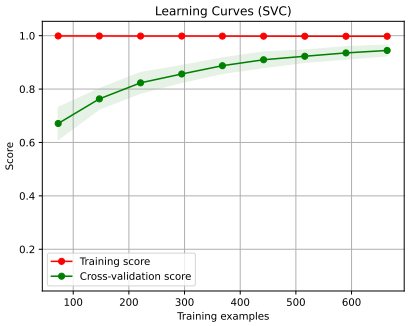
After the text has been extracted, processed, and parsed into the CSV file, the data is ready to be split into training and test sets and start the training process for the classification methods. The more data is used, the more accurate the methods become. Hence, it has been decided to start by preparing 500 documents from each class, invoice, and non-invoice. The test began by visualizing the Learning-Curve performance for each method for the selected amount of data. The shape of the data that has been used is **(1000, 2)** meaning that there are 1000 rows and two columns, namely a column that represents text and a column that represents it's label.



(a) Learning curve of LR.



(b) Learning curve of MNB.



(c) Learning curve of SVC.

Figure 5.1: Learning curves of the selected methods.

The Learning curves presented in Figure 5.1, indicate that it requires about 600 - 700 documents for the methods to perform well and achieve the required accuracy level.

5.2 Results

As more data becomes available, three different tests will be performed. These three tests will have a data set that varies in size. For the first test, a small dataset will be used, just to know if a small amount of documents is enough for the methods to perform well, according to what Section 5.1.1 shows. Since more data will almost always increase the accuracy of the models, two other tests with extended data sets will be performed.

5.2.1 Small Dataset

Figure 5.1 indicates that around 700 documents are needed to achieve good results. Hence, data of shape **(2 000, 2)** with balanced classes have been tested. The size of the training set is **1 400**, and the size of the test set is **600**. Table 5.1 shows information on the execution time for creating the vector representation. It also shows the execution time for training and testing each classification method.

Method	Training time	Testing time	Creation of the vector representation
LR	12.06 s	0.01 s	6.41 s
MNB	1.20 s	0.01 s	6.41 s
SVC	47.50 s	1.31 s	6.41 s

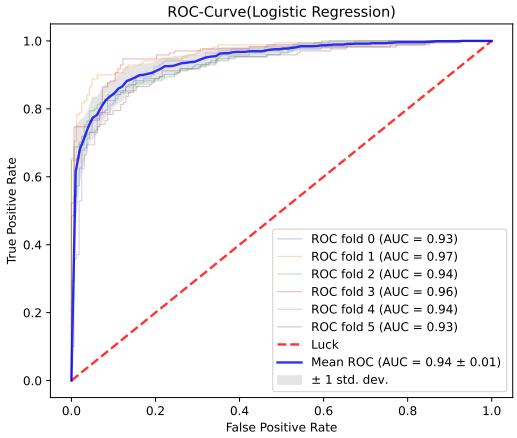
Table 5.1: Execution time of training, testing and creating vector representation for the small dataset.

The evaluation process for this amount of data is presented by Table 5.2, which presents evaluation metrics such as accuracy, F1-score, precision, and recall. Furthermore, 5-fold cross-validation has been used to indicate the best possible result and also to select the best hyper-parameters, as Table 5.2 shows.

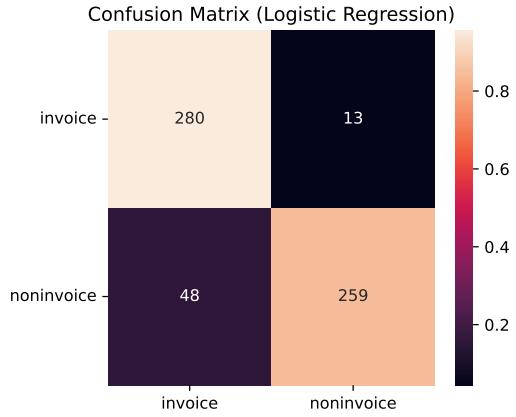
Method	H-parameters	Acc	F1	Precision	Recall
LR	{C: 100, max-iter: 1000, penalty: l2, solver: lbfgs}	0.89	0.90	0.90	0.90
MNB	{alpha: 0.1, fit-prior: True}	0.84	0.85	0.85	0.85
SVC	{C: 100, Kernel: linear}	0.90	0.90	0.91	0.90

Table 5.2: The result of 5-fold cross-validation-based average metrics for the small dataset.

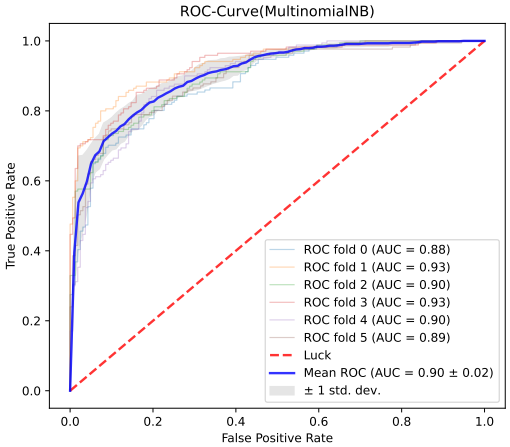
In addition, ROC-curves have been plotted for each method to show how their performance changes when the average value is assigned to the invoice or non-invoice class. Figure 5.2 shows that each of the subfigures has 6 curves, five of the curves correspond to the five iterations of the cross-validation, and the sixth curve indicates their mean value. The dashed line shows how the performance of the methods has been at a random distribution. Although the amount of data that has been used is rather small, the methods have still given good results, according to their ROC-curves and confusion matrices. This may indicate even better results for future tests as more data will be used.



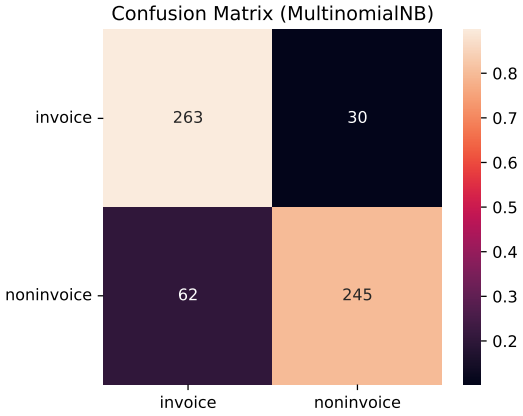
(a) ROC-curve of LR.



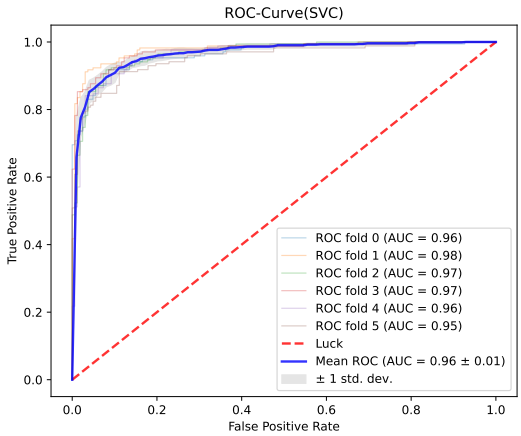
(b) Confusion matrix of LR.



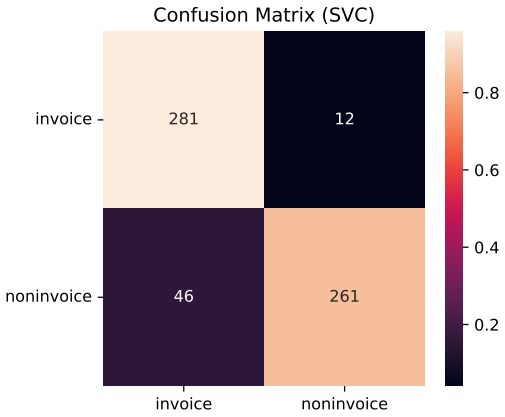
(c) ROC-curve of MNB.



(d) Confusion matrix of MNB.



(e) ROC-curve of SVC.



(f) Confusion matrix of SVC.

Figure 5.2: ROC-curves and confusion matrices of the selected methods for the small data set.

5.2.2 Medium Dataset

When it comes to this test, everything that has been done in the previous test will be repeated but with more data. The data that has been used for this test has a shape of **(11 981, 2)**. The size of the training data is **8 386**, and the size of the test data is **3 595**.

Method	Training time	Testing time	Creation of the vector representation
LR	82.34 s	0.02 s	34.63 s
MNB	4.61 s	0.02 s	34.63 s
SVC	334.50 s	9.71 s	34.63 s

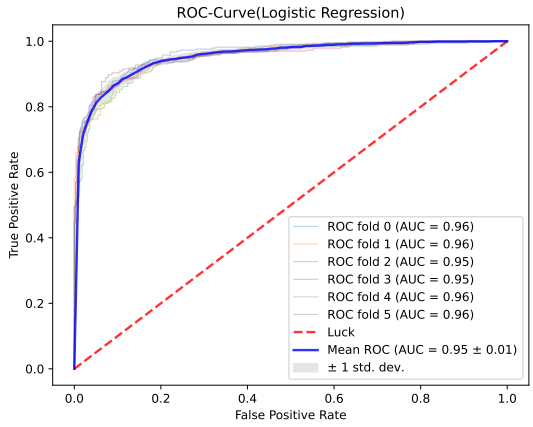
Table 5.3: Execution time of training, testing and creating vector representation for the medium dataset.

Table 5.3 indicates that more time is needed to create the vector representation and to train the methods. It is noticeable that Support Vector Classification (SVC) is relatively slow to train compared to the other two methods, despite using a linear kernel. This is a clear sign of how time-consuming the training of the SVC method will be if a large data set is used.

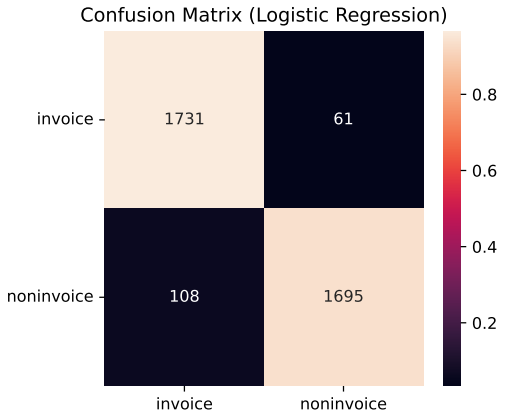
Method	H-parameters	Acc	F1	Precision	Recall
LR	{C: 100, max-iter: 1000, penalty: l2, solver: lbfgs}	0.95	0.95	0.95	0.95
MNB	{alpha: 0.1, fit-prior: True}	0.91	0.91	0.91	0.91
SVC	{C: 100, Kernel: linear}	0.95	0.95	0.95	0.95

Table 5.4: The result of 5-fold cross-validation-based average metrics for the medium dataset.

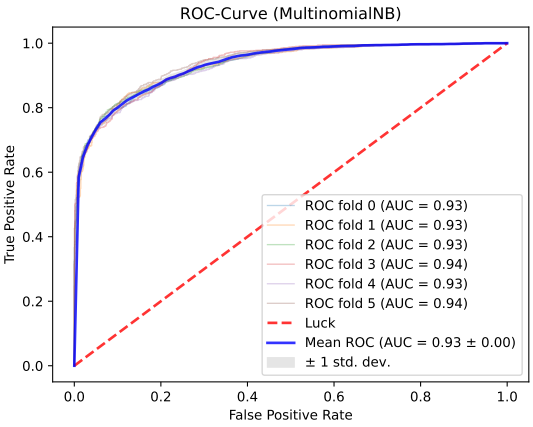
Table 5.4 shows that the best accuracy is obtained by LR and SVC. In addition, Figure 5.3 shows that the confusion matrix for both methods is almost the same. However, the same figure shows that the performance across all possible classification thresholds for SVC is better than LR by 2 percentage points according to their AUCs.



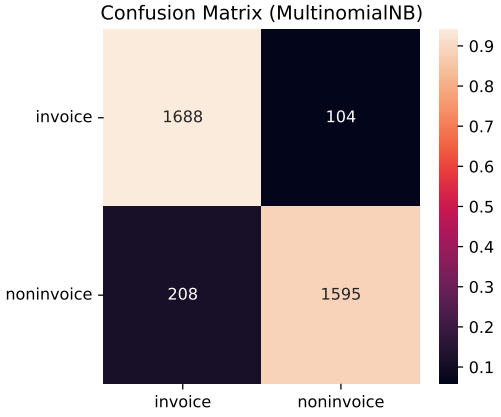
(a) ROC-curve of LR.



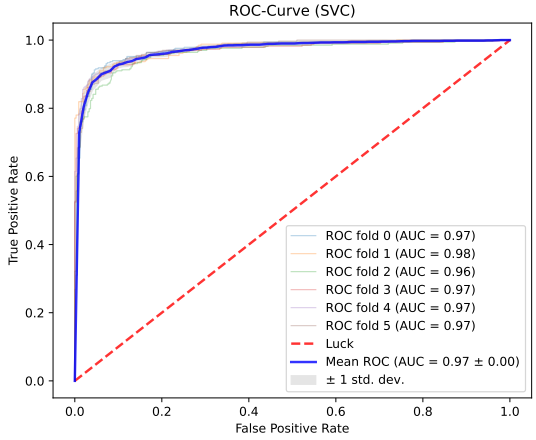
(b) Confusion matrix of LR.



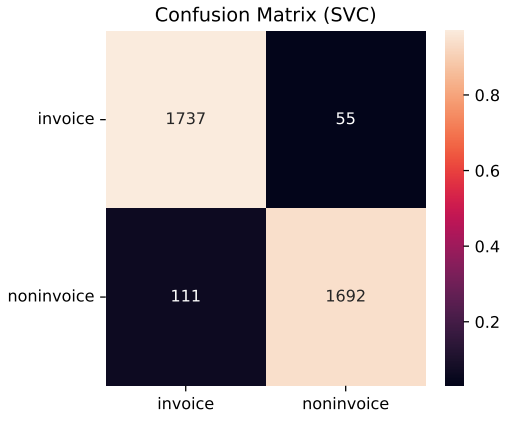
(c) ROC-curve of MNB.



(d) Confusion matrix of MNB.



(e) ROC-curve of SVC.



(f) Confusion matrix of SVC.

Figure 5.3: ROC-curves and confusion matrices of the selected methods for the medium data set.

5.2.3 Large Dataset

The previous results showed an increased accuracy when more data was used. For this reason, it is reasonable to use more data. Hence, for the last test it has been chosen to use a data set of shape **(288 872, 2)** with **202 210** as training data and **86 662** as test data. The data is balanced, that is, the number of documents for the two classes is equal. Table 5.5 below shows the execution time to create the TF-IDF vector and the time it took to train and test the methods. SVC took approximately 50 hours to train, which is rather slow compared to the other two methods.

Method	Training time	Testing time	Creation of the vector representation
LR	1 938.90 s	0.19 s	606.67 s
MNB	105.63 s	0.56 s	606.67 s
SVC	182 392.91 s	1 881.94 s	606.67 s

Table 5.5: Execution time of training, testing and creating vector representation for the large dataset.

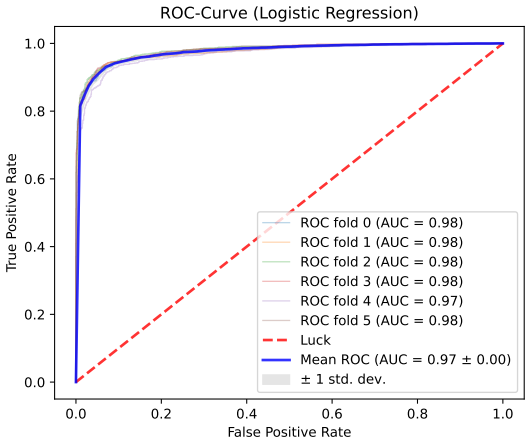
For this test, it has been chosen to manipulate the dimension of the encoded documents vector, and this is related to deciding which terms to include when the vector will be created. For this, various parameters have been manipulated when creating the vector representation. These parameters are max_df and min_df. The max_df parameter is used for removing words that appear very frequently. On the other hand, min_df is used to remove words that occur infrequently. These parameters determine which terms "words" should be used when creating the encoded documents vector. Setting max_df = 0.8 means that all words that appear in more than 80% of the documents are ignored. The same goes for min_df, setting it to 0.1 means that all words that appear in less than 10% of the documents are ignored.

Because the data is rather large and experimenting with different values for these parameters is time-consuming, especially for SVC, it has been decided to only test specific values of these parameters just to vary the dimension of the vector. Table 5.6 shows the evaluation metrics for each of the methods with varying vector dimensions.

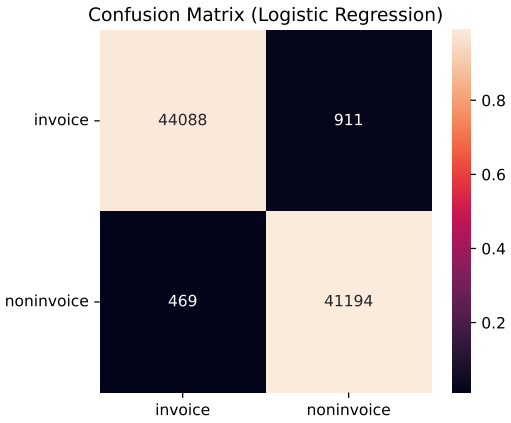
max-df	min-df	Vector-dim	Method	Acc	F1	Precision	Recall
1.0	0.1	113	**LR**	**0.94**	0.93	0.94	0.94
			MNB	0.91	0.91	0.91	0.91
			SVC	0.94	0.92	0.92	0.96
0.8	0.05	767	**LR**	**0.95**	0.94	0.92	0.97
			MNB	0.92	0.93	0.93	0.93
			SVC	0.98	0.98	0.98	0.98
1.0	1.0	7 362 791	**LR**	**0.98**	0.98	0.99	0.98
			MNB	0.93	0.92	0.90	0.96
			SVC	0.98	0.98	0.99	0.98

Table 5.6: The result of 5-fold cross-validation-based average metrics for the large dataset.

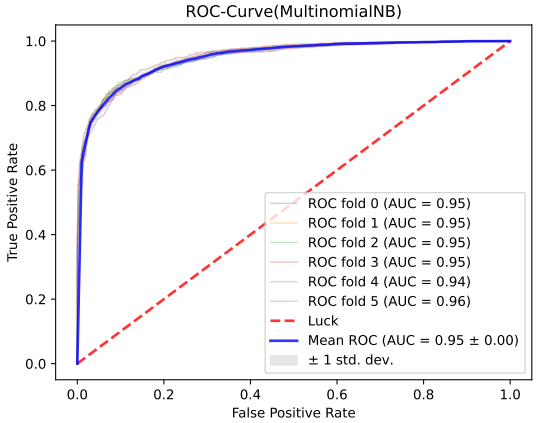
Table 5.6 shows that higher values are obtained in the evaluation matrices when using a bigger dimension of the vector. Based on this, it has been decided that the methods that have been trained with full dimension of the vector representation of words (7 362 791) are more suitable to test on real data. Figure 5.4 shows the ROC-curves and the confusion matrices of the methods that have been trained with the whole vector dimension. More discussion about this test will be covered in Section 6.2.3.



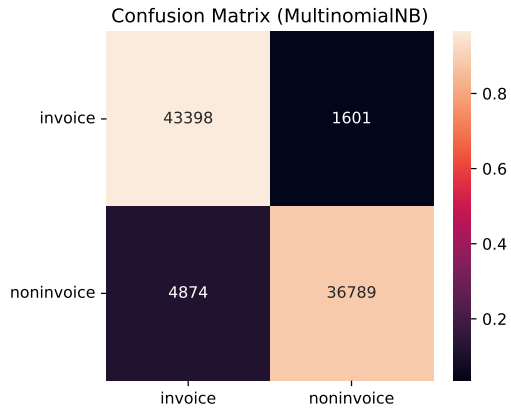
(a) ROC-curve of LR.



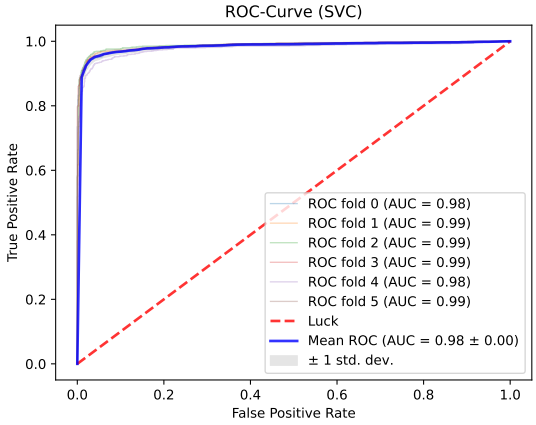
(b) Confusion matrix of LR.



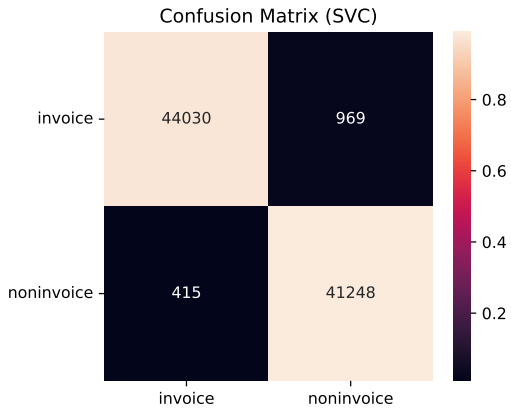
(c) ROC-curve of MNB.



(d) Confusion matrix of MNB.



(e) ROC-curve of SVC.



(f) Confusion matrix of SVC.

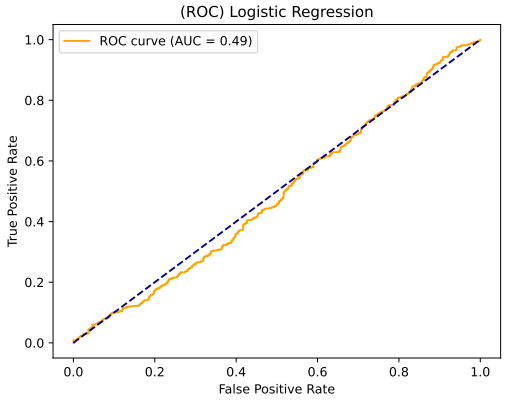
Figure 5.4: ROC-curves and confusion matrices of the selected methods for the large data set.

5.3 Additional Test on Doc2Vec

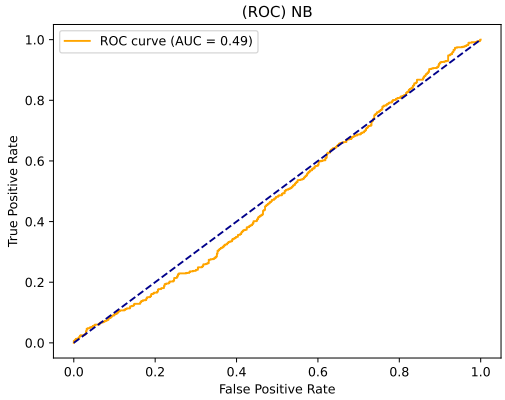
When it to Doc2Vec, it has been decided to use all available data. Hence, the data set of shape **(288 872, 2)** with **202 210** as training data and **86 662** as test data has been used to perform this test. The test results are not as detailed as in previous tests, but the most important thing that should be pointed out has been included. Table 5.7 and Figure 5.5 shows the results obtained by the methods. Due to the low values of the evaluation metrics obtained by Doc2Vec, no further tests will be performed.

Method	Acc	F1	Precision	Recall
LR	0.50	0.51	0.43	0.51
MNB	0.48	0.50	0.38	0.51
SVC	0.51	0.50	0.41	0.51

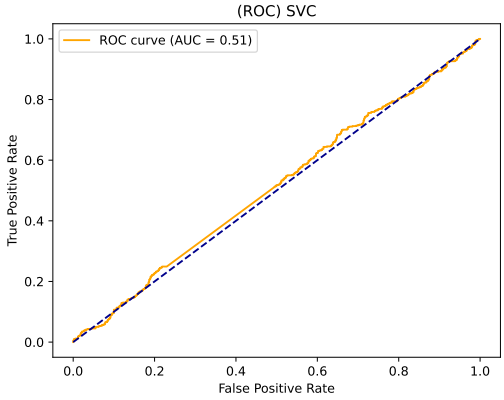
Table 5.7: The result of 5-fold cross-validation-based average metrics for the doc2vec-based test.



(a) ROC-curve of LR.



(b) ROC-curve of MNB.



(c) ROC-curve of SVC.

Figure 5.5: The ROC-curves of the selected methods for the doc2vec-based test.

5.4 Test on Real Data

The data that has been tested was the first version that has been gathered and manually categorized by a supervisor at Medius, as has been discussed in Section 4.1.

The first category "Financial Documents" only contains invoice documents. The data was supposed to have only invoice documents, as they were manually processed. With LR, 29 documents were classified as non-invoice documents. Out of these 29 documents, 20 documents were actually non-invoice documents. For the Image category, 2 images were classified as invoices, which has been indicated to be true. This category should include images that do not represent invoices, such as advertising, company logos, and empty white pages. However, further manual checking of the documents that do not belong to the corresponding categories has not been performed. This will be taken in more detail in Chapter 6.

However, according to Table 5.8 LR method seems to perform better than the other methods when testing on real data.

Category	Number of documents	Method	Correctly classified documents	Incorrectly classified documents	Classification time in seconds
Financial-Documents	617	**LR**	608	9	147
		MNB	**617**	0	630
		SVC	613	4	246
Images	44	**LR**	**44**	0	17
		MNB	32	12	55
		SVC	41	3	14
Terms Conditions	42	**LR**	**37**	5	17
		MNB	9	33	52
		SVC	2	40	21
Other Documents	119	**LR**	**83**	36	44
		MNB	16	103	137
		SVC	17	102	46

Table 5.8: The obtained result of testing the methods on unseen documents.

Chapter 6

Discussion

This chapter aims to discuss the choice of the methods as well as the results that have been obtained. Various critical points of view that aim to highlight the further improvements of this thesis will also be discussed. Furthermore, this chapter aims to draw a conclusion as to whether or not the questions of this thesis have been answered. Future work and improvements will also be presented. Finally, this chapter concludes by reflecting on what has been learned during the time period of the thesis.

6.1 Method Selection

As mentioned earlier, various methods for representing text "feature extraction" and classifying documents have been implemented and tested. Some of these methods have been chosen as the most suitable methods. The motivations for selecting these methods are discussed below.

6.1.1 Selection of Vectorizing Methods

In Chapter 2, various methods for text representation have been presented. Simple methods that do not take into account syntactic and semantic properties between the words as well as more complex methods have been represented. These complex methods are more adapted and optimized for classification tasks such as spam detection, news genre, detection of fake news, etc. This is simply because these tasks require sentence analysis to capture semantics and syntactic properties between words. This thesis aims to represent individual and independent words that make the classification possible without considering syntactic and semantic properties between these words, as has been briefly mentioned in Section 2.6. Since the proposed task of this thesis is considered as a short-text task which in turn does not always observe the syntax and its language-related properties [66], it is unnecessary to apply methods that take these language properties into account. In addition, it has been discussed in the same chapter that other methods can be adopted and applied to classify invoices, such as layout and template-based methods. Due to the significant number of vendors and thus different invoice templates, it becomes more unlikely that there will be invoices with similar layouts or templates. In addition, the invoices may be poorly scanned, which makes it difficult to capture details and patterns so that correct classification is performed.

To not complicate the task and adopt complex methods that are not optimized to solve this relatively non-complex task or adopt template-based methods that require a vast amount of templates to achieve the desired result, the choice of a simple and more appropriate method becomes more reasonable. This method is TF-IDF, and although it is simple compared to other methods that have been presented, such as Word2Vec, Doc2Vec, and BERT, it has provided higher accuracy. This clearly shows that simple techniques can provide more adequate results if suitable parameters are used, enough data is presented for the generalization, and the data is preprocessed appropriately.

However, it has been decided to test Doc2Vec in order to demonstrate the inefficiency of complex

methods, which require enormous optimization processes and a huge amount of data, for solving the proposed task for the thesis.

6.1.2 Selection of Classification Methods

A total of three methods have been selected to classify the documents and compare their results according to the mentioned evaluation matrices. These methods are:

1. Logistic Regression
2. Multinomial Naïve Bayes
3. Support Vector Machine

These three methods have been chosen based on their high prevalence in previous research [67]–[69]. Previous research indicates that these methods are the most widely used and provide the most promising results in various classification tasks.

In addition, it can also be observed that in previous research, these methods have been tested on different text-based data sets with varying amounts of data. However, it has been challenging to find studies that performs tests on these methods on unstructured "invoice" text-based data sets. Therefore, it will be interesting to test these models on such a data set and see the results.

Since the feature extraction method that has been used in this thesis is TF-IDF, it becomes logical to use methods that work and are based on such a feature extraction method [3]. However, it is equally interesting to test these methods on Doc2Vec-based document vectors and compare their results with TF-IDF's.

6.2 Result Analysis

sThree different tests have been performed and presented in Section 5.2, where different data sets that vary in size have been used. As mentioned, all datasets have a balanced number of classes.

6.2.1 Small Dataset

The first test has been related to what Learning curves have shown. According to Figure 5.1, not much data has been needed for the models to perform well and obtain a high result. Hence, the first test has been performed on a small amount of data, namely 2000 documents. The result from the first test indicates that good results in terms of accuracy have been obtained. As observed, it has not been required that much time to train the models and create the vector representation of words, as Table 5.1 shows. In addition, Table 5.2 shows that higher values of the evaluation matrices have also been obtained, despite the limited amount of data. Furthermore, Figure 5.2 shows that the methods have achieved a high performance and can easily manage this task, according to what their ROC-curves and Confusion matrices show.

In summary, LR has demonstrated an excellent ability to cope with this task, as it takes less time to train, and it can easily distinguish the invoice documents from the non-invoice documents, according to Figure 5.2b.

6.2.2 Medium Dataset

As access to more data is available, and to meet the requirements of this dissertation, where over 90% accuracy must be obtained, more data have been tested. For this test, a data set of 11 981 documents has been performed. Remarkably, the SVC test requires more time to train when more data is used in the learning process. According to Table 5.3, the SVC has taken 5.5 minutes to be trained, which can be considered relatively slow compared to the other two methods. In addition, it has been observed that even more accuracy and performance can be obtained if more data is used, according to Table 5.4 and Figure 5.3.

Furthermore, according to the Confusion matrices shown in Figure 5.3, it has been relatively easy for the methods to distinguish invoice documents from non-invoice documents, as the invoice documents are less misclassified. This may be because more frequent terms have been included in the vector representation of words. Therefore, it has been decided to vary the dimension of the vector representation of words for the next test, to determine how the number of frequent terms affects the result.

To summarize this test, using more data increased the value of the evaluation matrices and the performance of the methods. Still, LR has demonstrated its ability to cope with this task regarding less training time and high classifying performance. However, SVC performs almost the same with more time required for training.

6.2.3 Large Dataset

As already discussed in Section 5.2.3, more data and varied dimensions of the vector representation of words have been used. It turns out that the accuracy and the other evaluation matrices become higher when more terms have been included in the vector representation of words, especially for LR, as Table 5.6 indicates. In addition, it indicates that the methods can easily distinguish invoice documents from other documents, which is the desired goal of this thesis. Figure 5.4 shows the ROC-curves and the Confusion matrices of the methods. The Confusion matrix for LR shows that only 911 documents have been incorrectly classified as non-invoice documents, which corresponds to only 2% of the invoice documents. For SVC, there is not much difference from LR in terms of the correct classified invoice documents. However, MNB indicates a slightly skewed result. In contrast, the confusion matrix for MNB shows that for non-invoice documents, 4874 documents are misclassified as invoice documents, which corresponds to approximately 12% of the non-invoice documents. The same figure shows that the AUC of LR and SVC is slightly better than MNB in terms of performance, which is true of what their Confusion matrices indicate.

To summarize this test, the methods' accuracy and performance become higher when more terms have been vectorized and included in the vector representation. In terms of time, hardware can be a limiting factor in how different methods perform. It has been shown that SVC requires 50.5 hours to be trained, even though a linear kernel, that does not require more resources, has been used [70]. Again, LR seems to cope with this task very easily, as LR has it easier to classify the documents compared to the other methods.

6.2.4 Doc2Vec-based Test

The result of this test shows that Doc2Vec is not a suitable method or needs to be optimized even more carefully to achieve slightly higher results. The result is not acceptable and does not meet the thesis requirements. Therefore it is not logical to use it to classify unseen documents. Other research has indicated that Doc2Vec is not a suitable method for classifying short-text documents in the context of financial documents [71]. More concluding details have been discussed in Section 6.2.6.

6.2.5 Real Data Test

The test on the real data has indicated that LR has performed better than MNB and SVC. When it comes to the first category "Financial Documents", LR has managed to correctly classify 609 of 617 invoice documents, which corresponds to 99% of the documents in this category. On the other hand, both MNB and SVC have performed better in classifying this category. However, it may be suspected that some overfitting has occurred with these two methods, given their result in classifying the other categories.

For the second category "Images", LR has managed to correctly classify all images, which corresponds to 100% of all files. SVC missed three images and classified them as invoices, which is surprising compared to the result it got for classifying the first category.

When it comes to the third category, "Terms and Conditions" which is the penultimate category in the difficulty level of classification, LR has still shown an excellent ability to distinguish these

documents. The challenge with these documents in this category is that they contain a lot of text, which means that the probability that the text includes the words used by the methods to distinguish invoice documents is high. LR has missed only five documents, which is still considered to be an appropriate result.

The last category is "Other documents" which contain financial documents that do not represent invoices but have a very similar layout. For a normal human being, it will take some time to realize that such documents do not actually represent invoice documents. Therefore this category is the most complex and most challenging to be classified. LR has shown an excellent ability to classify the documents, where only 36 of 119 documents have been classified as invoices which is incorrect. However, this category has not been verified that it actually only contains non-invoice documents. Since some incorrectly categorized documents have been discovered in the first and second categories, as has been discussed in Section 5.4. There are still possibilities that this category contains invoice documents. Unfortunately, no manual checking has been performed to verify this.

6.2.6 Summary

In summary, it has been indicated that LR with TF-IDF-based vectorizer performs better than other methods for classifying documents, which has been confirmed in this study [72]. In other previous studies [71], [73], it has been indicated that LR together with TF-IDF achieves higher accuracy than other methods, Which has also been confirmed by this thesis given the obtained results. In the same studies, it has been discussed that Doc2Vec is not an appropriate method to use for short-text tasks, because Doc2Vec needs thousands of terms in each document to get sufficient context. It also seems to be true, given the results obtained by Doc2Vec. This thesis has confirmed that Doc2Vec is not a suitable method or needs to be optimized to be used for short-text tasks, including invoice classification.

6.3 Critical Analysis

Although LR performed well, especially on the real test data, there are still a few things to keep in mind and must be highlighted. Comparing MNB and SVC's results with LR's can provoke many reconsiderations. Possible reconsiderations are discussed below, which can probably increase the reliance of the methods on classifying financial documents.

6.3.1 Optical Character Recognition

Since the data set contains PDF files with a text-layer, images, and scanned PDF files without a text-layer, it was a must to handle these files separately when it comes to text extraction. Medius has its own OCR-engine, but for the public data set, two Python libraries have been used to extract the text as it has already been presented in Chapter 4. The problem that can arise is how well these OCR-engines recognize the text. As it was a part of the thesis delimitations, no optimization method for the rotation of PDF pages or images that facilitates text detection has been implemented. In addition, no image preprocessing method that makes the text clearer has been implemented either. Many questions about how well these OCRs recognize and extract the text can arise. Some notable cases when LR was tested on "Finacial Documents", were that some documents were misclassified even though they were actually invoices. The problem was that OCR failed to recognize and extract the word "invoice". The word was recognized as "nvoice". However, this can be avoided with various methods for spelling correction, such as Edit Distance [74].

6.3.2 Preprocessing

As has been discussed before, the text must be pre-processed to achieve a high result. In this step, various choices would have affected the desired result. One of these choices is the Stemming method. Since different languages have been detected in the data set, as it has mentioned before in Section 4.1, it has been challenging to use stemming methods for all detected languages. Since the majority of the languages in the data set were English, only the English Stemming analyzer method has been used. Other detected languages have not been taken into account when it comes to the Stemming analyzer. In addition, no assumptions have been made as to how other languages have been affected by the English stemming analyzer. One way to solve this problem is to translate all non-English texts into English. This can be implemented by, e.g., Google Translate API. However, how well Google translates the text must also be considered, as even small spelling mistakes can affect the result. This can relate to how OCR-engines extract the text, as even slight misspellings can affect translation accuracy.

6.3.3 The Dataset

Since the public data set RVL-CDIP used for the second class "noninvoice" may not be related to the business documents that Medius receives from their customers, it would be better to prepare non-invoice documents from their mixed data set, which has been presented in Section 4.1, and use them instead of the public data set. This will probably increase the accuracy of distinguishing between what are considered to be invoice documents and what are considered to be other documents.

Another thing that might be worth thinking about is again the data set. When LR was tested on the manually categorized folder "Financial Documents", which should contain only invoice documents, a large amount of these documents were classified as non-invoices. After a meticulous manual checking on these documents, it was correctly ruled that these were non-invoice documents. This raised considerable doubt about the confidence of the provided data by Medius, as the data set was supposed to only have invoice documents. Hence, a thorough verification to ensure that no non-invoice documents are included may be required.

One last thing that is important to mention is the language variation among the documents. The plurality of languages for the document in which they are written makes it difficult to classify the documents written in the minority of detected languages. In addition, the data set is very unbalanced when it comes to the language distribution, where the majority goes to English, followed by Swedish. This places significant restrictions on the minority of languages to be classified correctly. Furthermore, all data used as non-invoices were written in English. In my opinion, this gives bad effects on non-invoices written in other languages, as they were not a part of the training process.

6.3.4 Hyper-parameters Tuning

As it has been presented in Chapter 4, the classification methods have several hyper parameters that can have different values. In order to optimize the methods so that the desired result is achieved, these parameters must be set to the most appropriate value. This can be obtained by trying different values for each parameter and using some sort of Cross-validation method. For Naive Bayes, there were two hyper parameters, so trying different values on the two parameters will not be so time-consuming. However, SVC and LR have 15 hyper parameters and optimizing each of them will be very time-consuming, especially for SVC. In addition, Gridsearh(), which has been used to optimize the hyperparameters and perform k-fold cross-validation, is computationally heavy.

Only some of these hyper parameters have been tested, as presented in Section 5.2. Optimizing more hyper parameters will be time-consuming, but will probably improve the method's performance and results.

6.4 Conclusion

This thesis aims to analyze and evaluate various machine learning classification methods in terms of their accuracy and efficiency for the task of financial document classification. Three classification methods have been analyzed and tested with different data sets and with various settings including hyper-parameters, different dimensions of the vector representation of words and cross-validation. Although many critical points of view have been highlighted in Section 6.3, the goal of demonstrating the ability to classify financial documents has been achieved. In addition, all proposed requirements have been satisfied regarding the accuracy and time required to classify the documents. Furthermore, the additional requirements have also been fulfilled. By looking at the obtained results that LR got in Section 5.4, it can be observed that the method's precision is 96%, as only 50 documents of 822 were misclassified. The same figure shows that no image has been misclassified by LR. As a result, all additional requirements have been also satisfied.

To sum up, all prerequisites have been fulfilled and the implementation of classifying invoice documents is considered successful. The Logistic Regression together with TF-IDF is determined to be the overall most appropriate method out of the other tested methods. In addition, Doc2Vec suffers to provide a good result because the data set is not customized and sufficient for the method to work well

6.5 Social and Ethical Aspects

The motivation of this thesis was to help Medius to optimize their invoice workflow. The social benefit that this thesis offers, is that money and time can be saved for classifying invoices that are handled digitally. Actually, it is not just Medius that can take advantage of the work in this thesis, as they want to automate their invoice workflow, but also other companies that strive for the same thing. It can also be helpful for other industries that want to handle their documents in an automated way, as the work in this thesis can be optimized to achieve the desired goal.

However, the classification of financial documents in a supervised manner will always be the basis for most companies, although an automated classification is considered to be the dominant possibility in the future. In fact, there is always a risk that financial documents such as invoices will be misclassified. This will have significant consequences for the company, especially if large amounts of documents are incorrectly classified. However, the ethical aspects are considered enormous, not only for misclassified or incorrectly handled documents, but for employees who are currently working on manual classification of the documents. From the employees' perspective, automated classification will be the harsh reality, as their job will be considered redundant. From society's perspective, this can be considered to be a burden that will lead to increased unemployment. This can also be considered to be a politically related question. In fact, how society should adapt to the increased automation and digitization is most relevant to highlight and discuss.

6.6 Future Studies and Improvements

There are different contexts that can be applied for text classification and not just handling financial documents. Therefore, the research done in this thesis can be further developed.

Further development of this research can begin by taking into account the points of view that have been discussed in critical analysis as they can positively contribute to obtaining better results.

A further development process that can help gain even more accuracy is using the positions of the text fields in the learning process. Actually, it has been planned to implement this process in this study, but this plan came a bit late. In addition, OCR-engines must be analyzed and evaluated to ensure that they actually detect and register the correct positions of the text fields, which also will take time. However, it is essential to pay extra attention when implementing this process. It's primarily about not allowing the classification methods to put more confidence in

the positions, which makes the methods completely dependent on them more than focusing on solving the actual task. Since it is less likely that the invoices that have been used in the learning process and new invoices have similar layouts, the positions may vary and in the end the invoices may be incorrectly classified.

Furthermore, testing more classification methods to perhaps improve the result can be a future improvement. In addition, it may be interesting to try to adopt and optimize some of the methods that have been presented in Section 2.6. Moreover, combining several algorithms together can be exciting to try and analyze. This can be done with a library called mlxtend [75].

6.7 Reflection on Own Learning

The desired benefit of what this master's degree thesis gave me is that I have planned, designed, implemented, and evaluated everything from scratch during the time period of the thesis. Planning how this thesis will be carried out, what methods will be used, and how everything will be evaluated took me quite a bit more time than was needed. But the time I spent on the planning gave very good results. Everything that was scheduled in my Gantt chart was precisely fulfilled in time.

I have always found NLP very interesting to delve into, especially after taking a course on information retrieval. This thesis has been an excellent educational opportunity to learn about NLP and text classification tasks. Moreover, the implementation of this thesis was carried out through different phases, which resulted in expanding my knowledge and skills. The thesis gave me a great opportunity to increase my skills in a variety of topics. I have done research on the following topics: machine learning, NLP, character recognition, and computer vision. This required me to read many scientific articles and a huge amount of research on these topics. However, this introduced many new ideas, methods, problems, and reassessments. Sometimes it became difficult to see what was right or wrong, which method was appropriate to choose, and why it was appropriate. These questions became easy to answer because of my increased comprehension.

In addition, the weekly meetings with my supervisors gave a very good impression for organizing and structuring my workflow. Moreover, it gave me an excellent impression for optimizing and utilizing the working time in the best way, and also increased my communication skills. I am most proud of the ability to argue the choice of the proposed methods, which helps to increase my self-confidence. This particular part of this thesis "choosing the most suitable methods" was a challenge for me, as it required careful research and insight into how the proposed methods worked. Moreover, this thesis has helped me stay informed about current methods and developments state of NLP and text classification.

Bibliography

[1] Ikonomakis, M, Kotsiantis, S., and Tampakas, V, "Text classification using machine learning techniques.", *WSEAS transactions on computers*, vol. 4, no. 8, pp. 966–974, 2005.

[2] Khurana, D., Koli, A., Khatter, K., and Singh, S., "Natural language processing: State of the art, current trends and challenges", *arXiv preprint arXiv:1708.05148*, 2017.

[3] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D., "Text classification algorithms: A survey", *Information*, vol. 10, no. 4, p. 150, 2019.

[4] Ramos, J. *et al.*, "Using tf-idf to determine word relevance in document queries", in *Proceedings of the first instructional conference on machine learning*, Citeseer, vol. 242, 2003, pp. 29–48.

[5] Kim, J., Jang, S., Park, E., and Choi, S., "Text classification using capsules", *Neurocomputing*, vol. 376, pp. 214–221, 2020.

[6] Singh, S., "Natural language processing for information extraction", *arXiv preprint arXiv:1807.02383*, 2018.

[7] Hotho, A., Nürnberger, A., and Paass, G., "A brief survey of text mining", *LDV Forum*, vol. 20, pp. 19–62, 2005.

[8] Khan, A., Baharudin, B., Lee, L. H., and Khan, K., "A review of machine learning algorithms for text-documents classification", *Journal of advances in information technology*, vol. 1, no. 1, pp. 4–20, 2010.

[9] Azam, M., Ahmed, T., Sabah, F., and Hussain, M. I., "Feature extraction based text classification using k-nearest neighbor algorithm", *IJCSNS Int. J. Comput. Sci. Netw. Secur*, vol. 18, no. 12, pp. 95–101, 2018.

[10] Chen, J., Huang, H., Tian, S., and Qu, Y., "Feature selection for text classification with naïve bayes", *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 5432–5435, 2009, ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2008.06.054`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0957417408003564`.

[11] Jović, A., Brkić, K., and Bogunović, N., "A review of feature selection methods with applications", in *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, Ieee, 2015, pp. 1200–1205.

[12] Witten, I. H., Frank, E., and Hall, M. A., *Data mining: Practical machine learning tools and techniques*, 2011.

[13] Benesty, J., Chen, J., Huang, Y., and Cohen, I., "Pearson correlation coefficient", in *Noise reduction in speech processing*, Springer, 2009, pp. 1–4.

[14] Wu, J., Xuan, Z., and Pan, D., "Enhancing text representation for classification tasks with semantic graph structures", *International Journal of Innovative Computing, Information and Control (ICIC)*, vol. 7, no. 5, pp. 2689–2698, 2011.

[15] Huang, E., Socher, R., Manning, C., and Ng, A., "Improving word representations via global context and multiple word prototypes", in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 873–882. [Online]. Available: `https://aclanthology.org/P12-1092`.

[16] Sitikhu, P., Pahi, K., Thapa, P., and Shakya, S., "A comparison of semantic similarity methods for maximum human interpretability", in *2019 artificial intelligence for transforming business and society (AITB)*, IEEE, vol. 1, 2019, pp. 1–4.

[17] Kim, H. K., Kim, H., and Cho, S., "Bag-of-concepts: Comprehending document representation through clustering words in distributed representation", *Neurocomputing*, vol. 266, pp. 336–352, 2017.

[18] Barathi Ganesh, H., Anand Kumar, M, and Soman, K., "Vector space model as cognitive space for text classification", *Notebook for PAN at CLEF*, 2017.

[19] Kalra, S., Li, L., and Tizhoosh, H. R., "Automatic classification of pathology reports using tf-idf features", *arXiv preprint arXiv:1903.07406*, 2019.

[20] Levy, O. and Goldberg, Y., "Dependency-based word embeddings", in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, pp. 302–308.

[21] Rossiello, G., Basile, P., and Semeraro, G., "Centroid-based text summarization through compositionality of word embeddings", in *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*, 2017, pp. 12–21.

[22] Mikolov, T., Chen, K., Corrado, G., and Dean, J., "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781*, 2013.

[23] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J., "Distributed representations of words and phrases and their compositionality", *Advances in neural information processing systems*, vol. 26, 2013.

[24] Le, Q. and Mikolov, T., "Distributed representations of sentences and documents", in *International conference on machine learning*, PMLR, 2014, pp. 1188–1196.

[25] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[26] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation", *arXiv preprint arXiv:1609.08144*, 2016.

[27] Bishop, C. M. and Nasrabadi, N. M., *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.

[28] Learned-Miller, E. G., "Introduction to supervised learning", *I: Department of Computer Science, University of Massachusetts*, p. 3, 2014.

[29] Aggarwal, C. C. and Zhai, C., "A survey of text classification algorithms", *Mining Text Data*, p. 163, 2012.

[30] Rish, I. *et al.*, "An empirical study of the naive bayes classifier", in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, 2001, pp. 41–46.

[31] McCallum, A., Nigam, K., *et al.*, "A comparison of event models for naive bayes text classification", in *AAAI-98 workshop on learning for text categorization*, Citeseer, vol. 752, 1998, pp. 41–48.

[32] Kibriya, A. M., Frank, E., Pfahringer, B., and Holmes, G., "Multinomial naive bayes for text categorization revisited", in *Australasian Joint Conference on Artificial Intelligence*, Springer, 2004, pp. 488–499.

[33] Boser, B. E., Guyon, I. M., and Vapnik, V. N., "A training algorithm for optimal margin classifiers", in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.

[34] Haykin, S *et al.*, "Neural networks and learning machines.[sl] pearson upper saddle river", *NJ, USA*, vol. 3, 2009.

[35] Bishop, C. M. *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[36] Alsmadi, M. k., Omar, K. B., Noah, S. A., and Almarashdah, I., "Performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks", in *2009 IEEE International Advance Computing Conference*, 2009, pp. 296–299. DOI: 10.1109/IADCC.2009.4809024.

[37] Werbos, P. J., "Backpropagation through time: What it does and how to do it", *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[38] Engelbrecht, A. P., *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[39] Lindholm, A., Wahlström, N., Lindsten, F., and Schön, T. B., *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[40] Janssens, A. C. J. and Martens, F. K., "Reflection on modern methods: Revisiting the area under the roc curve", *International journal of epidemiology*, vol. 49, no. 4, pp. 1397–1403, 2020.

[41] Perlich, C., *Learning curves in machine learning*. 2010.

[42] Kieckbusch, D. S., Geraldo Filho, P., Di Oliveira, V., and Weigang, L., "Scan-nf: A cnn-based system for the classification of electronic invoices through short-text product description", 2021.

[43] Noce, L., Gallo, I., Zamberletti, A., and Calefati, A., "Embedded textual content for document image classification with convolutional neural networks", in *Proceedings of the 2016 ACM Symposium on Document Engineering*, 2016, pp. 165–173.

[44] Manaswi, N. K., "Rnn and lstm", in *Deep Learning with Applications Using Python*, Springer, 2018, pp. 115–126.

[45] Kim, Y., "Convolutional neural networks for sentence classification", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Aug. 2014. DOI: 10.3115/v1/D14-1181.

[46] Zhou, C., Sun, C., Liu, Z., and Lau, F., "A c-lstm neural network for text classification", *arXiv preprint arXiv:1511.08630*, 2015.

[47] Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks", *Advances in neural information processing systems*, vol. 28, 2015.

[48] Zhu, B., Wu, X., Yang, L., Shen, Y., and Wu, L., "Automatic detection of books based on faster r-cnn", in *2016 third international conference on digital information processing, data mining, and wireless communications (DIPDMWC)*, IEEE, 2016, pp. 8–12.

[49] Sun, Y., Mao, X., Hong, S., Xu, W., and Gui, G., "Template matching-based method for intelligent invoice information identification", *IEEE access*, vol. 7, pp. 28 392–28 401, 2019.

[50] Paalman, J., Mullick, S., Zervanou, K., and Zhang, Y., "Term based semantic clusters for very short text classification", in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 2019, pp. 878–887.

[51] Naseem, U., Razzak, I., Musial, K., and Imran, M., "Transformer based deep intelligent contextual embedding for twitter sentiment analysis", *Future Generation Computer Systems*, vol. 113, pp. 58–69, 2020.

[52] Schuster, M. and Paliwal, K. K., "Bidirectional recurrent neural networks", *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[53] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[54] Wang, R., Cheng, C., Meng, Y., Sun, Y., Yang, J., and Gui, G., "Lightweight deep learning model for invoice image classification", 2020.

[55] Raschka, S., *Python machine learning*. Packt publishing ltd, 2015.

[56] Shinyama, Y, *Pdfminer. six: Python pdf parser and analyzer san francisco: Github inc., 2018. accessed 2020 jul 20*.

[57] Hoffstaetter, S, Bochi, J, Lee, M, Kistner, L, Mitchell, R, Cecchini, E, Hagen, J, Morawiec, D, Bedada, E, and Akyüz, U, *Pytesseract 0.3. 8. python software foundation*, 2021.

[58] Millstein, F., *Natural language processing with python: natural language processing using NLTK*. Frank Millstein, 2020.

[59] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.*, "Scikit-learn: Machine learning in python", *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[60] McKinney, W. *et al.*, "Pandas: A foundational python library for data analysis and statistics", *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[61] Hunter, J. D., "Matplotlib: A 2d graphics environment", *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[62] Harley, A. W., Ufkes, A., and Derpanis, K. G., "Evaluation of deep convolutional nets for document image classification and retrieval", in *International Conference on Document Analysis and Recognition (ICDAR)*.

[63] Rehurek, R. and Sojka, P., "Gensim–python framework for vector space modelling", *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[64] Nguyen, Q. H., Ly, H.-B., Ho, L. S., Al-Ansari, N., Le, H. V., Tran, V. Q., Prakash, I., and Pham, B. T., "Influence of data splitting on performance of machine learning models in prediction of shear strength of soil", *Mathematical Problems in Engineering*, vol. 2021, 2021.

[65] Prabhu, A., Dognin, C., and Singh, M., "Sampling bias in deep active classification: An empirical study", *arXiv preprint arXiv:1909.09389*, 2019.

[66] Hua, W., Wang, Z., Wang, H., Zheng, K., and Zhou, X., "Short text understanding through lexical-semantic analysis", in *2015 IEEE 31st International Conference on Data Engineering*, IEEE, 2015, pp. 495–506.

[67] Tsangaratos, P. and Ilia, I., "Comparison of a logistic regression and naïve bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size", *Catena*, vol. 145, pp. 164–179, 2016.

[68] Thangaraj, M. and Sivakami, M, "Text classification techniques: A literature review", *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 13, pp. 117–135, Jan. 2018. DOI: 10.28945/4066.

[69] Sebastiani, F., "Machine learning in automated text categorization", *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[70] Hsu, C.-W., Chang, C.-C., Lin, C.-J., *et al.*, *A practical guide to support vector classification*, 2003.

[71] Wang, Y., Zhou, Z., Jin, S., Liu, D., and Lu, M., "Comparisons and selections of features and classifiers for short text classification", in *Iop conference series: Materials science and engineering*, IOP Publishing, vol. 261, 2017, p. 012 018.

[72] Pranckevičius, T. and Marcinkevičius, V., "Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification", *Baltic Journal of Modern Computing*, vol. 5, no. 2, p. 221, 2017.

[73] Daga, I., Gupta, A., Vardhan, R., and Mukherjee, P., "Prediction of likes and retweets using text information retrieval", *Procedia Computer Science*, vol. 168, pp. 123–128, 2020.

[74] Navarro, G., "A guided tour to approximate string matching", *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[75] Raschka, S., "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack", *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018. DOI: 10.21105/joss.00638. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638.